NAVAL POSTGRADUATE SCHOOL
Monterey, California

DISSERTATION

AERODYNAMIC DESIGN USING
PARALLEL PROCESSORS

by

LT Stephen C. Brawley

September 1993

Dissertation Supervisor: Garth Hobson

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
		Dissertation, Dec 1991 to Sept '9

4. TITLE AND SUBTITLE
Aerodynamic Design Using Parallel Processors

5. FUNDING NUMBERS

6. AUTHOR(S)
Brawley, Stephen C.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Naval Air Systems Command
AIR - 53 011
Washington, D.C. 20361

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

An airfoil design technique has been developed which decreases the computational processing time by more than an order of magnitude when optimizing aerodynamic performance. The practicality of airfoil design using parallel processors and Navier-Stokes flow solvers has been demonstrated.

Typically, an airfoil is designed to meet certain criteria based upon its aerodynamic performance at set flight conditions. If an optimization technique is used for airfoil design, the shape of the airfoil is varied, and the aerodynamic perfomance of numerous airfoil geometries are evaluated using computational fluid dynamics. Multiple aerodynamic performance evaluations require the vast majority of computational processing time used in airfoil design optimization.

14. SUBJECT TERMS		15. NUMBER OF PAGES
Aerodynamic Design	Parallel Processing	153
Optimization	Runge-Kutta Euler Solver	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited

NSN 7540-01-280-5500

Standard Form 298 (Rev 2-89)
Prescribed by ANSI Std 239-18
298-102

Efficient Euler and Navier-Stokes flow solvers which take advantage of the vector processing capabilities of modern processors are used with optimization schemes for internal and external aerodynamic design. Processors of the iPSC/860 Intel hypercube parallel computer are utilized to simultaneously evaluate the performance of numerous airfoil shapes. The utilization of multiple processors in parallel greatly decreases the computational processing time and increases the efficiency of the optimization design process.

Approved for public release; distribution is unlimited.

Aerodynamic Design Using Parallel Processors

by

Stephen C. Brawley
Lieutenant, United States Navy
B.S., United States Naval Academy, 1984
M.S., Naval Postgraduate School, 1991

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN AERONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September, 1993

Author: Stephen C. Brawley
Stephen C. Brawley

Approved by:

Daniel J. Collins
Professor of Aeronautics

Oscar Biblarz
Professor of Aeronautics

Harold A. Titus
Professor of Electrical Engineering

Beny Neta
Professor of Mathematics

Garth V. Hobson
Professor of Aeronautics
Dissertation Supervisor

Approved by: Daniel J. Collins
Daniel J. Collins, Chairman, Department of Aeronautics and Astronautics

Approved by: Richard S. Elster
Richard S. Elster, Dean of Instruction

ABSTRACT

An airfoil design technique has been developed which decreases the computational processing time by more than an order of magnitude when optimizing aerodynamic performance. The practicality of airfoil design using parallel processors and Navier-Stokes flow solvers has been demonstrated.

Typically, an airfoil is designed to meet certain criteria based upon its aerodynamic performance at set flight conditions. If an optimization technique is used for airfoil design, the shape of the airfoil is varied, and the aerodynamic performance of numerous airfoil geometries are evaluated using computational fluid dynamics. Multiple aerodynamic performance evaluations require the vast majority of computational processing time used in airfoil design optimization.

Efficient Euler and Navier-Stokes flow solvers which take advantage of the vector processing capabilities of modern processors are used with optimization schemes for internal and external aerodynamic design. Processors of the iPSC/860 Intel hypercube parallel computer are utilized to simultaneously evaluate the performance of numerous airfoil shapes. The utilization of multiple processors in parallel greatly decreases the computational processing time and increases the efficiency of the optimization design process.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

Funding for this work was provided through the 1992 Vice Admiral Wilkinson Fellowship sponsored by the Naval Air Systems Command and the Naval Postgraduate school . Computer facilities essential for this research were provided by the National Aerodynamic Simulation laboratory located at NASA Ames.

There are not enough words and paper to thank everyone sufficiently for their contributions to this dissertation, so I will briefly thank a few . First, I thank the United States Navy, the Naval Postgraduate School and the Naval Air Systems Command for giving me the opportunity and resources to complete this research.

In particular, I would like to thank my advisor Mr. Tom Lawrence of the Naval Air Systems Command and CDR Wade Duym of the Naval Postgraduate School for their efforts which made this research possible. Furthermore, I am indebted to Mr. Augustus Verhoff and Mr. Dave Stookesberry of the McDonnell Douglas Corporation for their guidance and support in many areas. I want to thank my dissertation supervisor and friend Professor Garth Hobson for the inspiration and assistance he has given me. It has been my pleasure to know and to work with all these gentlemen.

Finally, I thank my wonderful family, Kimberly, Austin and Cameron, who always gives me an endless amount of support and love.

I. INTRODUCTION

A. BACKGROUND

Computational fluid dynamics or CFD has become a valuable engineering tool in both aerodynamic analysis and design. CFD has evolved in the last 20 years and is presently undergoing extensive development in many areas such as turbulence modeling, hypersonics and the utilization of more powerful computers leading to parallel computation.

1. CFD As A Design Tool

Computational fluid dynamics involves the numerical solution of the governing equations of fluid flow and can be used for both design and analysis purposes. Design methods can be classified as either inverse or optimization techniques. Inverse methods solve for the geometry directly based upon a prescribed pressure distribution. Optimization methods vary the geometry of an initial body until an objective function, typically based upon a pressure distribution or drag, is minimized. Inverse methods are generally faster but do not allow the designer as much flexibility in the design process.

The problem of designing an airfoil to match a desired pressure distribution was first addressed by Lighthill [1] in 1945. He solved the problem for incompressible potential flow around an airfoil by conformally mapping the profile into a unit circle.

Jameson [2] suggests regarding airfoil design as a control problem. A desired pressure distribution and an initial airfoil shape are selected. Similar to Lighthill's technique, mapping functions conformally map the shape of the airfoil into a unit

circle . Flow tangency conditions are set at the surface of the circle and free-stream conditions are set at the far-field boundaries. The Euler equations or the potential flow equations are used to solve the flow-field. Potential flow solvers can only be used up to low transonic Mach numbers where insignificant entropy changes occur. The conformal mapping function is then varied in the optimization routine using calculus of variation techniques.

Giles and Drela [3] developed a two-dimensional inverse design code based upon the simultaneous solution of multiple streamtubes coupled through position and pressure on the streamline faces. The inviscid Euler equations are assembled and solved in conservative form. The airfoil surface or pressure distribution can be specified for either analysis or design. Viscous effects are included by displacing the surface streamline by the calculated displacement thickness around the airfoil. An intrinsic finite-volume grid is used to evaluate the finite-difference equations in which one family of grid-lines represents streamlines. The continuity and energy equations require a constant mass flow rate and constant enthalpy along each streamtube, and the momentum equation is simplified due to no mass flux across the streamlines. A Newton numerical method is used to solve the system of equations. Geometric constraints can be incorporated for design purposes by specifying the geometry for part of the airfoil and its pressure distribution for the remainder of the airfoil.

Campbell and Smith [4] have developed an optimization method to analyze the flow around an initial airfoil geometry and to modify the geometry based on differences between the calculated and target pressures. Geometric constraints are included in this method to ensure reasonable results. This predictor/corrector approach relates differences in the pressure field to changes on the airfoil surface.

The procedure predicts a change in the pressure distribution along the airfoil and relates this to the change in the curvature of the airfoil, which in turn is used to perturb the geometry. Many design iterations are required in transonic flow because small geometric changes result in large pressure changes near shocks. A significant advantage to this and other optimization methods is that they can be used in conjunction with various flow solvers; however, a new grid and a new flow-field solution are required after every geometric correction.

Bock [5] designed a transonic airfoil using the optimization code CONMIN (Constrained Function Minimization) developed by G. N. Vanderplaats [6]. A set of independent variables were defined which describe the airfoil geometry. The objective function to be minimized was defined as the wave drag coefficient evaluated by an Euler flow solver. In CONMIN, the gradient method is used to vary the independent variables.

Sanger [7] also used the CONMIN routine for optimization of compressor blade design. A potential flow solver with momentum-integral boundary layer calculations was used for performance evaluations based upon the suction side boundary layer separation point on the blade.

Kennelly [8] developed the optimization routine QNMDIF based upon a quasi-Newton method and compared the routine to CONMIN for utilization in airfoil design. QNMDIF provides the capabilities for central-difference approximations for the gradient and restart of the problem after a computer run. Kennelly [8 :p. 2] also points out the following characteristics relevant to airfoil design using optimization techniques:

1. Performance evaluations which include computational flow-field analysis dominate the required processing time in a design application.

2. The values of derivatives necessary for the optimization routine are approximated by finite differences and require numerous performance evaluations.

3. More design variables and more precise performance evaluations increase the processing time required in a design problem.

4. The designer should anticipate user intervention in the design process.

All of these points will be addressed in this dissertation.

2. Parallel Computers

Supercomputers are important to CFD because of the speed and storage capabilities that they bring to aerodynamic simulation which can supplement or sometimes replace experimental methods. Miel [9] observes that NASA and aircraft manufacturers routinely use supercomputers to calculate air flow over wings in new designs to eliminate weeks of wind tunnel testing on physical models.

Miel [9:p. 33] adds that an important trend in supercomputing is the emergence of parallel processors, and he divides high performance computers into three categories. The first category consists of course grain vector machines which include the Cray supercomputers. Moderately parallel machines such as the Intel hypercube with 128 relatively inexpensive processors constitute the second category. The third category of high performance computers consists of fine grain, massively parallel machines such as the Connection Machine CM-3. There is also a significant trend toward merging vector-parallel capabilities in supercomputing.

In Miel's report [9:pp.33-34], Peter Denning of NASA-Ames points out several reasons for the growing importance of parallel computers which include :

- Decreasing cost of processors

• The nonlinear relationship between a problem's size and the computer power needed to solve it, such as multiplying two n x n matrices requires n^3 operations

• The physical limit on the speed of a single processor is already being approached.

In general, the difficulty and cost of programming parallel computers have hindered their usefulness to CFD analysis and design; however, they can have a particularly important role in aerodynamic design using CFD and optimization methods. Computers have the capability of easily analyzing various geometries for comparison, which makes them a more practical design tool than experimental methods in many circumstances. Parallel processing machines can provide the additional capability of analyzing the performance of numerous geometries simultaneously for comparison in an optimization design routine. Therefore, numerous processors working on the same optimization problem can greatly increase the speed of the design process.

B. PURPOSE

Optimization methods for airfoil design have many advantages; however, their main disadvantage is the amount of computer time required for the design criteria to be optimized. Multiple CFD solutions are necessary to evaluate the performance of airfoils of various geometries and constitute the vast majority of computer processing time. Parallel optimization routines, when coupled with efficient flow solvers, can greatly reduce the computational time necessary to design an airfoil to match or optimize a desired aerodynamic performance.

In order to significantly speed up the design process, the time required for each flow-field calculations must be reduced. An efficient flow solver is required to quickly evaluate the aerodynamic performance associated with various airfoil geometries in a design application. Also, parallel processors can evaluate multiple airfoil geometries simultaneously to speed up the design process and increase the efficiency of the optimization routine.

The goal of this research is to decrease the time required for aerodynamic optimization design by utilizing efficient flow solvers and parallel processing machines. Sequential and parallel quasi-Newton and Newton optimization routines are evaluated in airfoil design test cases. The efficiency and relative speed of each is determined. In addition, the benefits and costs of using multiple parallel processors for the parallel optimization methods are discussed.

C. ORGANIZATION OF DISSERTATION

Airfoil optimization is a multi-disciplinary science, and this dissertation primarily discusses the importance of efficient flow solvers and the utilization of parallel processors in the design process. Chapter II discusses the governing equations applied in two-dimensional CFD. Chapter III presents the development of an explicit Euler flow solver used for airfoil design. Chapter IV describes the utilization of parallel processors in the optimization scheme, and Chapter V explains the utilization of the flow solver and optimization routine for airfoil design. Chapter VI presents the results of four airfoil design test cases and applications. Chapter VII presents the summary and conclusions of this research and suggests future work in this field.

II. GOVERNING EQUATIONS OF TWO-DIMENSIONAL COMPUTATIONAL FLUID DYNAMICS

A. FUNDAMENTAL EQUATIONS

The fundamental equations of fluid flow are based upon the following physical laws of conservation:

1. Conservation of Mass
2. Conservation of Momentum
3. Conservation of Energy

In addition, it is necessary to establish relationships between fluid properties to complete the set of equations.

1. Continuity Equation

The continuity equation is derived from the application of the Conservation of Mass applied to a fluid passing through an infinitesimal, fixed control volume. The net outflow of mass through the surface of a control volume must equal the net decrease of mass inside the control volume. For the simplified case of two-dimensional flow shown in Figure 2.1 with velocity u in the x direction and velocity v in the y direction, the following equation is formulated:

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x}(\rho u) + \frac{\partial}{\partial y}(\rho v) = 0 \tag{2.1}$$

2. Momentum Equation

The equation for the Conservation of Linear Momentum in fluid dynamics is obtained by applying Newton's second law to a fluid particle.

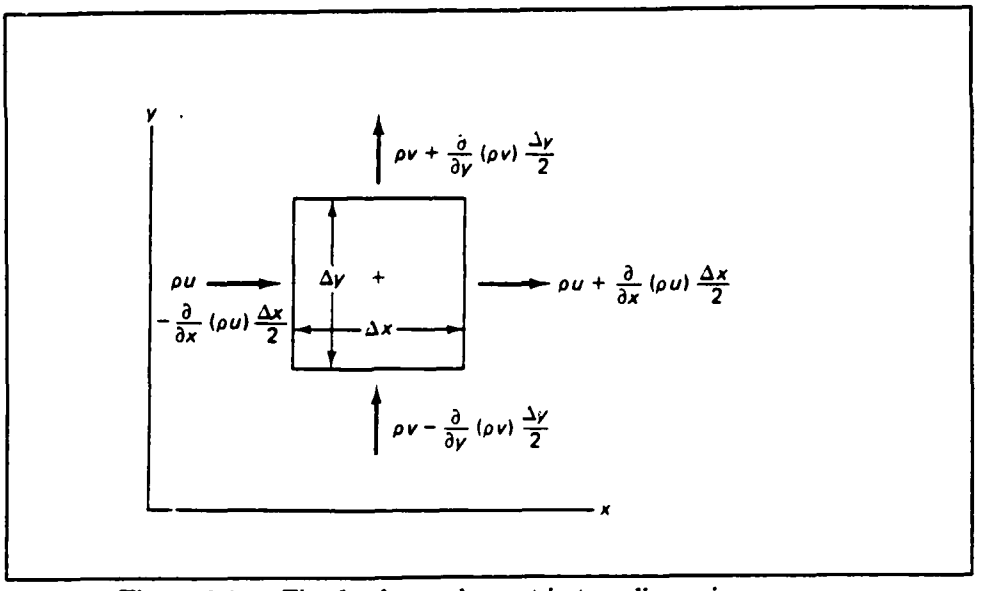

Figure 2.1 : Fixed volume element in two-dimensions

Using an inertial coordinate frame, the equation of conservation of linear momentum can be written :

$$\mathbf{F} = m\frac{d\mathbf{V}}{dt} \tag{2.2}$$

where **F** is the total force vector acting upon a fluid particle, m is the particle's mass, and **V** is the particle's velocity which in general is a function of time and position. Forces acting upon a fluid element include body forces such as gravity and shear stresses which act upon the surface of the element. Figure 2.2 illustrates the stresses acting upon a two-dimensional element. For the special case of a Newtonian fluid, the stress τ on a fluid element in all directions is directly proportional to the rate of strain Ω on the elements surface :

$$\tau = \mu\Omega \tag{2.3}$$

In an isotropic fluid, the proportionality constant μ known as the coefficient of viscosity is the same in all directions.

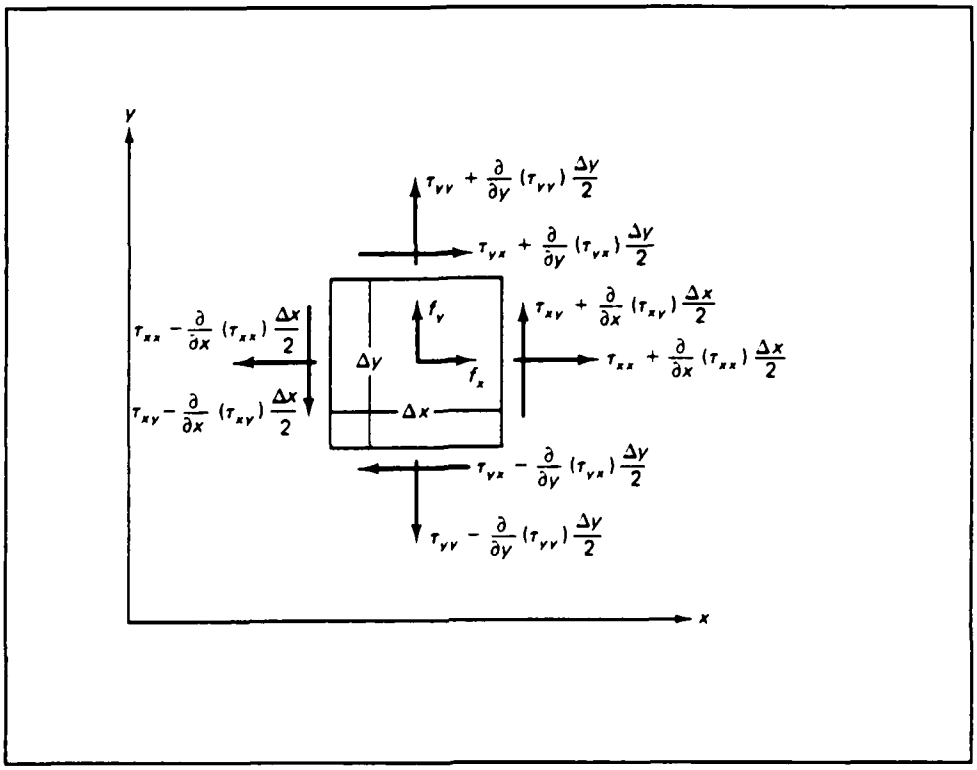

Figure 2.2 : Stresses acting on a two-dimensional fluid element

For a Newtonian, isotropic fluid in two-dimensional flow with a Cartesian coordinate system, the momentum equation can be written :

$$\rho\frac{\partial u}{\partial t} + \rho u\frac{\partial u}{\partial x} + \rho v\frac{\partial u}{\partial y} = \rho f_x - \frac{\partial p}{\partial x} + \frac{\partial}{\partial x}\left[\frac{2}{3}\mu\left(2\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right)\right] + \frac{\partial}{\partial y}\left[\mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)\right] \quad (2.4)$$

$$\rho\frac{\partial v}{\partial t} + \rho u\frac{\partial v}{\partial x} + \rho v\frac{\partial v}{\partial y} = \rho f_y - \frac{\partial p}{\partial y} + \frac{\partial}{\partial x}\left[\mu\left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right)\right] + \frac{\partial}{\partial y}\left[\frac{2}{3}\mu\left(2\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x}\right)\right] \quad (2.5)$$

Equations (2.4) and (2.5) are known as the Navier-Stokes equations. Here f refers to body forces per unit volume.

3. Energy Equation

The First Law of Thermodynamics in equation form can be written :

$$\int_1^2 \delta q + \int_1^2 \delta w = \int_1^2 de \quad (2.6)$$

It states that the heat transferred from the surroundings to a system plus the work done to the system by its surroundings equals the change of the energy of the system. Figure 2.3 illustrates the work done on an infinitesimal two-dimensional fluid element and its heat transfer terms. Applying Fourier's law of heat transfer by conduction and the results of the continuity and momentum equations for an isotropic two-dimensional fluid in a Cartesian coordinate system, the energy equation can be written :

$$\frac{\partial}{\partial x}(k\frac{\partial T}{\partial x}) + \frac{\partial}{\partial y}(k\frac{\partial T}{\partial y}) + 2\mu\left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2\right]$$
$$- \frac{2}{3}\mu\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)^2 + \mu\left[\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)^2\right] \qquad (2.7)$$
$$= \rho\frac{\partial \varepsilon}{\partial t} - \frac{dp}{dt}$$

Here T is the temperature, k is a coefficient of thermal conductivity, and ε is the specific internal energy of the fluid element.

B. EULER EQUATIONS IN VECTOR FORM

Aerodynamic problems involving high Reynolds number flows behave as inviscid flows over most of the flow-field except for a small but important region known as the boundary layer near the surface of the body. For initial design of aerodynamic bodies in high Reynolds number flows, the inviscid solution may give an adequate flow-field solution provided that viscous effects such as skin friction drag are not incorporated in the design criteria. For a more detailed design analysis, the boundary layer solution may be included after the initial inviscid design is determined.

The Euler equations consist of the continuity, momentum and energy equations without the viscous terms. In the formulation of the Euler equations, shear stresses caused by viscosity are ignored, and only normal stresses are considered.

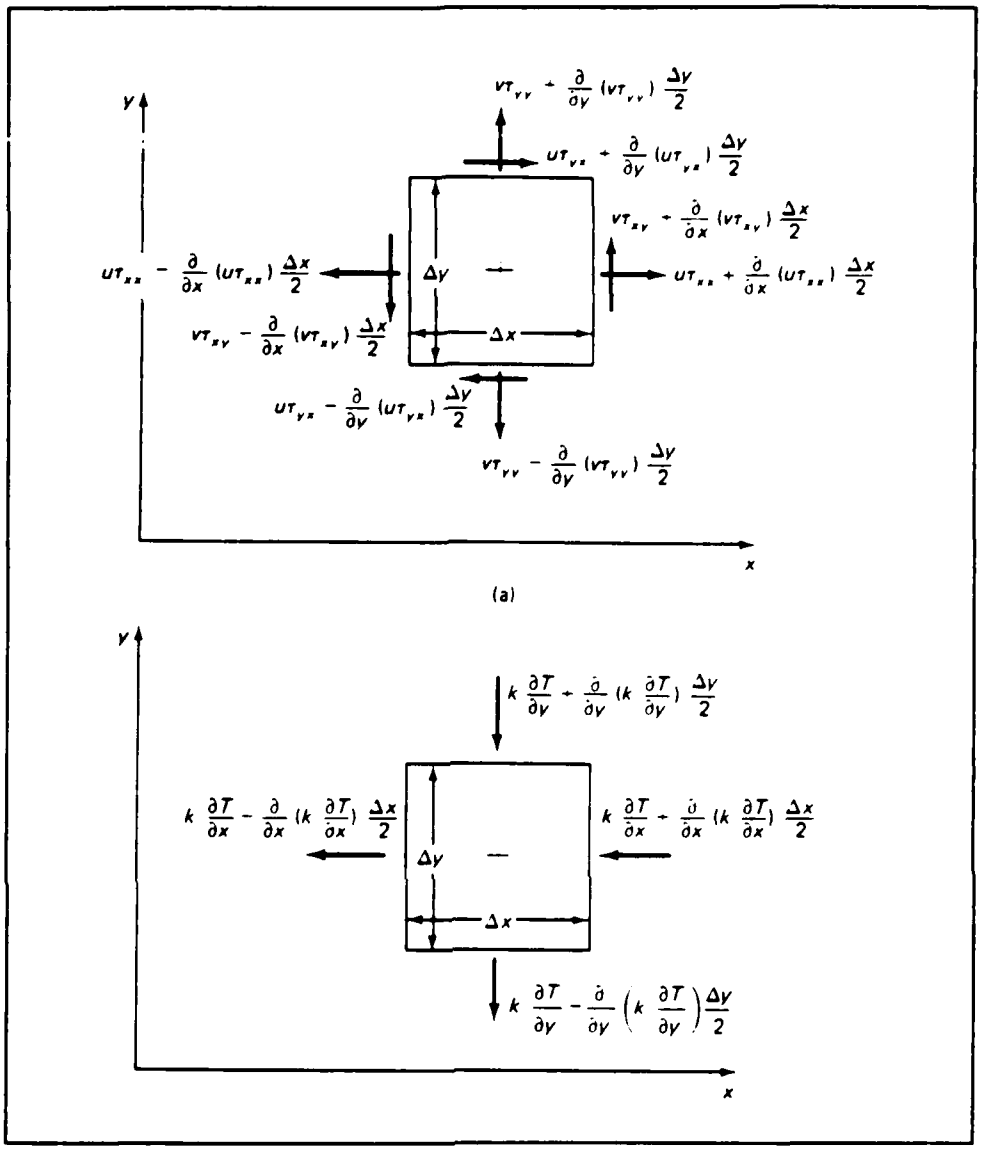

Figure 2.3 : a) Work done by stresses acting on two-dimensional fluid element
b) Heat transfer to a two-dimensional element

The Euler equations for two-dimensional flow in a Cartesian coordinate system can be written in vector form

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{E}}{\partial x} + \frac{\partial \mathbf{F}}{\partial y} = \mathbf{0} \tag{2.8}$$

where

$$\mathbf{Q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix}, \qquad \mathbf{E} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho u v \\ (e + p) u \end{bmatrix}, \qquad \mathbf{F} = \begin{bmatrix} \rho v \\ \rho u v \\ \rho v^2 + p \\ (e + p) v \end{bmatrix}$$

and e is total energy of the fluid per volume related to the internal energy by the relation :

$$e = \rho\left[\varepsilon + \frac{1}{2}\left(u^2 + v^2\right)\right] \tag{2.9}$$

C. EQUATIONS OF STATE

Since there are 4 equations and 5 unknowns, additional relationships between fluid properties are necessary to solve for the system of fundamental equations. The equation of state for a perfect gas supplies the following relationship between pressure, density, and temperature :

$$p = \rho R T \tag{2.10}$$

where R is the individual gas constant for the gas. Also for a perfect gas, the following relationship is held between specific internal energy ε and temperature :

$$\varepsilon = C_v T \tag{2.11}$$

where C_v is the specific heat at constant volume.

III. DEVELOPMENT OF AN EXPLICIT EULER FLOW SOLVER

The design of an aerodynamic body by an optimization routine requires numerous CFD flow-field evaluations in order to improve one or more selected performance criteria. Therefore, the vast majority of computer processing time in aerodynamic design with an optimization scheme is used solving the flow-field around various geometries. In this work, an efficient explicit Euler flow solver has been developed to reduce the computational time used in the CFD flow-field evaluations. The solution of the two-dimensional Euler equations in vector form is performed to estimate the pressure distribution around an airfoil in external flow.

A. COMPARISON OF FLOW SOLVER SCHEMES

Two basic types of schemes can be utilized in updating a flow-field in order to reach a CFD solution; namely, those based upon implicit or explicit algorithms. A Crank-Nicholson scheme contains a mixture of methods found in explicit and implicit schemes.

1. Implicit Schemes

A representative computational molecule for an implicit scheme is shown in Figure 3.1 for a one-dimensional case. The updated properties at a point depend upon the updated properties at adjacent points. Therefore, the properties of the entire flow-field are updated at the same time and require matrix inversion.

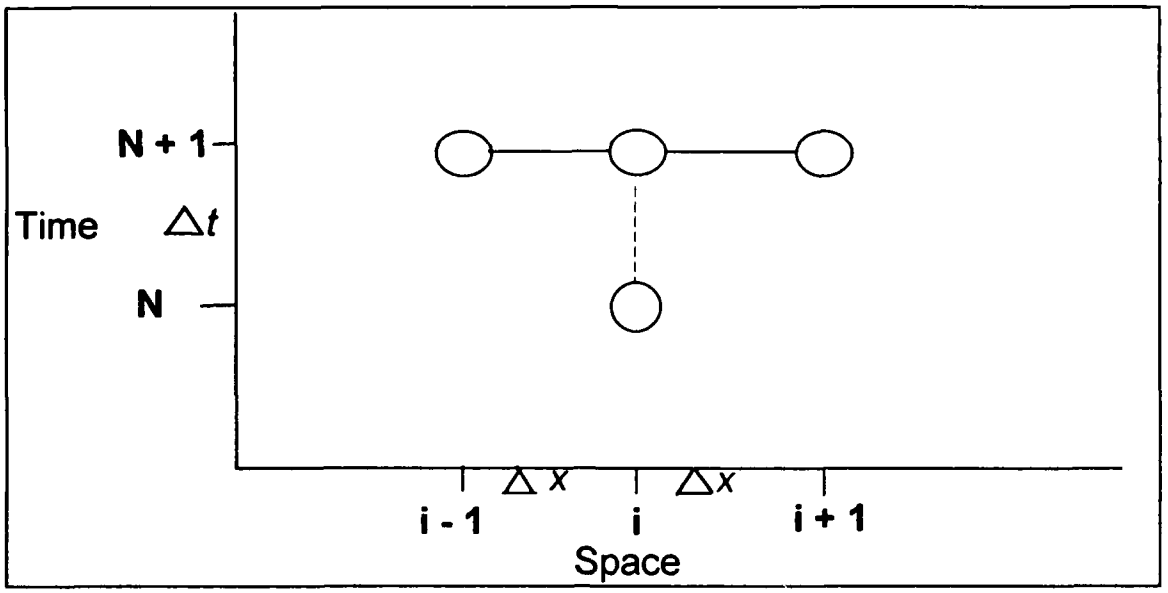

Figure 3.1 : Computational Molecule for an Implicit Scheme

Implicit two-dimensional flow solvers using an Alternating Direction Implicit (ADI) method evaluate the properties of the flow-field through alternating sweeps in both directions. These schemes solve for the updated properties of the entire flow-field in a single step and require the inversion of large tri-diagonal matrices. Flow-field properties are advanced in time to solve for their steady-state values.

Because all updated properties are solved for in the same step, there are no restrictions to the value of the time step applied to advance the flow-field solution. For implicit scheme flow solvers, large time steps can be used to estimate the steady-state solution to the governing equations, although the time steps can be too large and make the solution physically unrealistic. Also, an optimum time step exists for the fastest convergence to a steady-state flow-field solution. Because of the required matrix inversions, implicit schemes are not easily vectorizable for vector processors.

2. Explicit Schemes

The computational molecule for an explicit scheme is shown in Figure 3.2. The updated properties at a point depend upon the present properties at adjacent points. The properties of the entire flow-field are updated point by point, and matrix inversions are not required.

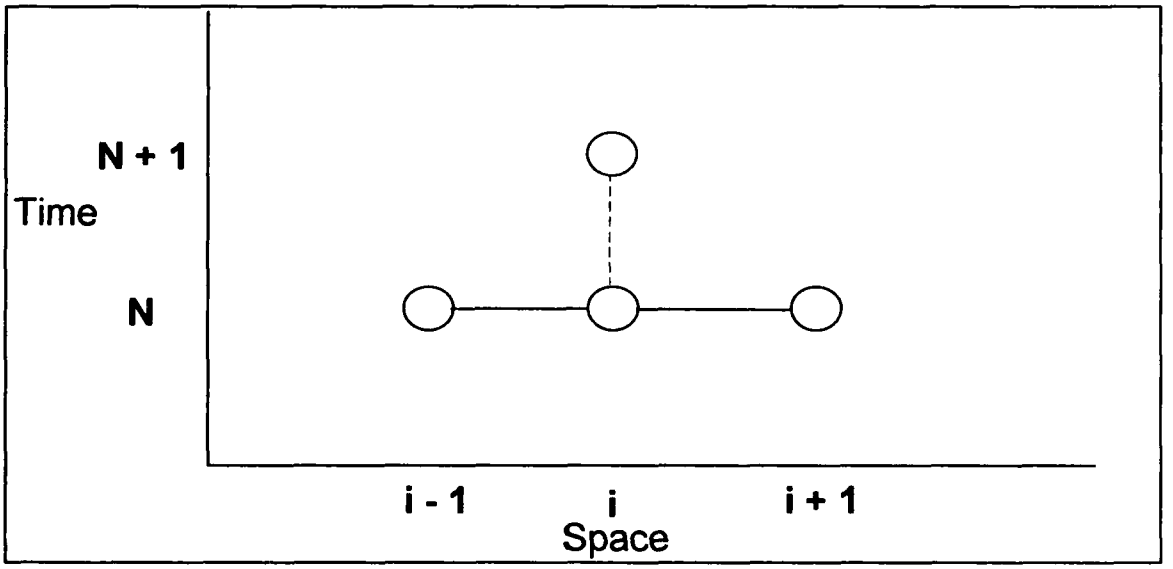

Figure 3.2 : Computational Molecule for an Explicit Scheme

In general, explicit flow solving schemes are easier to code than implicit schemes. Also, explicit codes are more easily vectorized than implicit ones and can execute faster on computers with vector processors. Because the properties in the flow-field are updated point by point, stability considerations restrict the maximum value of the time step utilized for explicit schemes. Merkle [10] suggests that multi-dimensional explicit scheme flow solvers are better suited to execute on processors with vectorization capabilities and on massively parallel computers than implicit scheme flow solvers. Furthermore, Merkle [10] notes that explicit schemes may be

the appropriate choice for unsteady CFD problems. The flow-field properties in unsteady CFD problems change over time, and small time steps are required to calculate the time-accurate solution.

3. Crank-Nicholson Schemes

The computational molecule for a Crank-Nicholson scheme is shown in Figure 3.3. The updated properties at a point depend upon the present and updated properties at adjacent points. The properties of the entire flow-field are updated at the same time which requires matrix inversion.

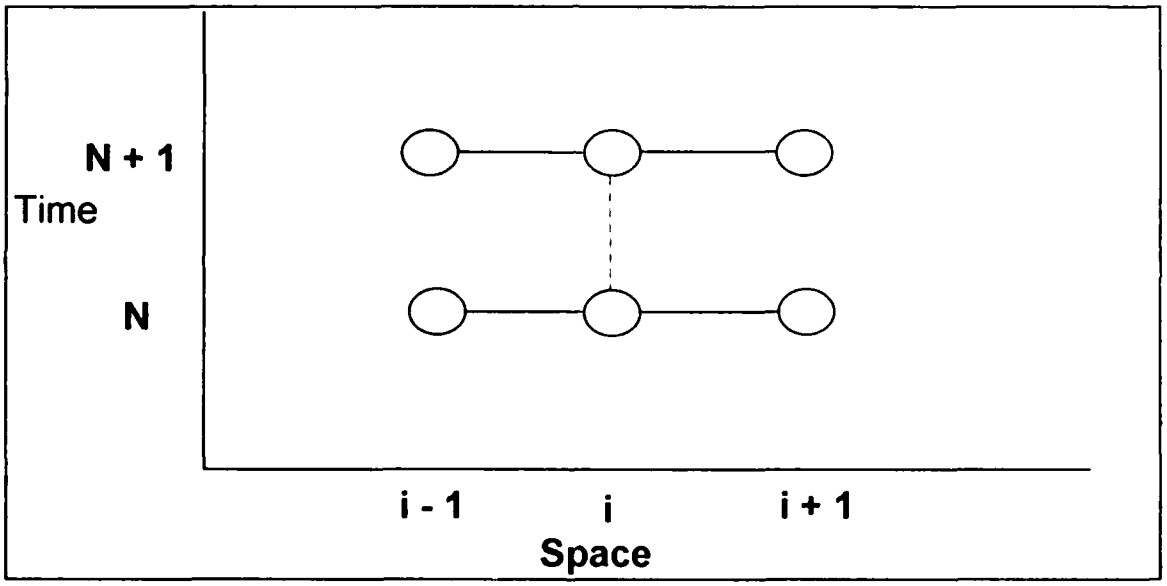

Figure 3.3 : Computational Molecule for a Crank-Nicholson Scheme

Crank-Nicholson scheme flow solvers are a mixture of both implicit and explicit schemes. Like implicit flow solvers, Crank-Nicholson flow solvers using the ADI method require inversions of large tri-diagonal matrices and are unconditionally stable. Crank-Nicholson schemes also are not easily vectorizable.

J. A. Ekaterinaris [11] has developed a two-dimensional Crank-Nicholson flow solver for high speed, high Reynolds number flow over an oscillating airfoil. The numerical integration uses a central-difference ADI method. Second-order dissipation is added for treatment of shocks and fourth-order dissipation is added for stability. The inviscid version of this scheme was modified to form a two-step Runge-Kutta scheme for the solution of the two-dimensional Euler equations describing steady flow.

B. IMPLEMENTATION OF THE TWO-STEP RUNGE-KUTTA SCHEME FOR AN EULER FLOW SOLVER

1. Formulation of two-step Runge-Kutta Scheme

In order to formulate the two-step Runge-Kutta algorithm, a second-order Taylor series in time for the vector **Q** in Equation (2.8) is taken :

$$\mathbf{Q}^{n+1} = \mathbf{Q}^n + \Delta t \frac{\partial \mathbf{Q}^n}{\partial t} + \frac{1}{2}\Delta t^2 \frac{\partial^2 \mathbf{Q}^n}{\partial t^2} + \text{H.O.T.} \tag{3.1}$$

An equivalent system of equations can be written to solve for the updated **Q** vector :

$$\mathbf{Q}^* = \mathbf{Q}^n + \alpha \Delta t \frac{\partial \mathbf{Q}^n}{\partial t} \tag{3.2}$$

$$\mathbf{Q}^{n+1} = \mathbf{Q}^n + \Delta t \frac{\partial \mathbf{Q}^*}{\partial t} \tag{3.3}$$

where the * represents an intermediate state in this scheme. The scalar α would equal 0.5 to exactly match the second-order Taylor series, which is equivalent to the

Lax-Wendroff method, and can be varied for stability and convergence considerations. The Euler equations are used in the form

$$\frac{\partial \mathbf{Q}}{\partial t} = -\left(\frac{\partial \mathbf{E}}{\partial x} + \frac{\partial \mathbf{F}}{\partial y}\right) \tag{3.4}$$

to solve for the time derivative $\partial \mathbf{Q} / \partial t$ terms in Equations (3.3) and (3.4). For the two-step Runge-Kutta scheme, an intermediate $\mathbf{Q}^*$ vector is evaluated throughout the flow-field in the first step, and the $\mathbf{Q}$ vector is updated the following step.

2. Calculation of Jacobian of Transformation and the Metrics

The C-type grid generated around the airfoil describes the physical domain in Cartesian coordinates. In order to calculate the flux terms in Equation (3.4) and to calculate dissipation terms, it is desirable to transform the irregularly spaced grid in the physical plane into a uniformly spaced rectangular grid in a computational plane. The Jacobians and metrics of this grid transformation are then needed to evaluate the flux and dissipation terms calculated in the computational plane.

For this problem, the (x,y) Cartesian coordinates in the physical plane are transformed to the (ξ,η) coordinates in the computational plane. Here, ξ and η are functions of x and y :

$$\xi = \xi(x,y)$$
$$\eta = \eta(x,y)$$

The differential expressions of the computational grid coordinates are written

$$d\xi = \xi_x dx + \xi_y dy \tag{3.5}$$

$$d\eta = \eta_x dx + \eta_y dy \tag{3.6}$$

where ξ_x, ξ_y, η_x, and η_y are the metrics. The Jacobian of the transformation is defined :

$$J = \begin{vmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{vmatrix} = \xi_x \eta_y - \xi_y \eta_x \tag{3.7}.$$

The following relationships for the metrics can then be derived:

$$\xi_x = J\, y_\eta \tag{3.8}$$

$$\xi_y = -J\, x_\eta \tag{3.9}$$

$$\eta_x = -J\, y_\xi \tag{3.10}$$

$$\eta_y = J\, x_\xi \tag{3.11}.$$

Equations (3.8) through (3.11) are used to determine the metrics for each point in the computational domain.

3. Initialization of the Flow-field Properties

Unless the problem is a restart of a previously run solution, similar properties throughout the flow-field are initialized to the same value. The density and pressure throughout the flow-field are initialized to values of 1.0 and the reciprocal of the ratio of specific heats for a perfect gas respectively. The velocity components and

energy per unit volume are calculated based upon the freestream Mach number, angle of attack of the airfoil, and equations of state.

4. Advancement of the Flow-field Solution

After the flow-field is initialized, the **Q** vector for each point in the flow-field is advanced in time using the two-step Runge-Kutta scheme until a convergence criterion is satisfied or the maximum number of iterations are completed.

The Courant number, or CFL, is selected to calculate the time step to advance the flow-field properties to their steady-state solution and is defined as

$$\mathrm{CFL} = \frac{\omega \Delta t}{l} \tag{3.12}.$$

The time step, Δt, is dependent upon a characteristic velocity, ω, and a characteristic length, l. The characteristic velocity is calculated in the computational plane, and the characteristic length is set to 1.0, which is the distance between adjacent points.

The characteristic velocity is determined from the eigenvalues of the Jacobian matrices of the 2-D Euler equations. Equation (2.8) in quasi-linear form is written

$$\frac{\partial \mathbf{Q}}{\partial t} + \mathbf{A}\frac{\partial \mathbf{Q}}{\partial x} + \mathbf{B}\frac{\partial \mathbf{Q}}{\partial y} = \mathbf{0} \tag{3.13},$$

where $\mathbf{A} = \frac{\partial \mathbf{E}}{\partial \mathbf{Q}}$ and $\mathbf{B} = \frac{\partial \mathbf{F}}{\partial \mathbf{Q}}$. **A** and **B** are the Jacobian matrices. Merkle [10] shows the Jacobian matrices are evaluated as

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{\gamma-3}{2}u^2 + \frac{\gamma-1}{2}v^2 & -(\gamma-3)u & -(\gamma-1)v & (\gamma-1) \\ -uv & v & u & 0 \\ \frac{-\gamma eu}{\rho} + (\gamma-1)u(u^2+v^2) & \frac{\gamma e}{\rho} - \frac{\gamma-1}{2}(3u^2+v^2) & -(\gamma-1)uv & \gamma u \end{bmatrix} \quad (3.14)$$

and

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ -uv & v & u & 0 \\ \frac{\gamma-3}{2}v^2 + \frac{\gamma-1}{2}u^2 & -(\gamma-1)u & -(\gamma-3)v & \gamma-1 \\ \frac{-\gamma ev}{\rho} + (\gamma-1)v(u^2+v^2) & -(\gamma-1)uv & \frac{\gamma e}{\rho} - \frac{\gamma-1}{2}(u^2+3v^2) & \gamma v \end{bmatrix} \quad (3.15)$$

where γ is the ratio of specific heats for a perfect gas. The eigenvalues of **A** and **B** are

$$\lambda_{\mathbf{A}} = (u, u, u+c, u-c) \quad (3.16)$$

and

$$\lambda_{\mathbf{B}} = (v, v, v+c, v-c) \quad (3.17)$$

where c is the speed of sound in the fluid.

For stability purposes, the characteristic velocity is based upon the largest eigenvalues of the **A** and **B** Jacobian matrices :

$$\omega = \sqrt{(u_{con}+c)^2 + (v_{con}+c)^2} \tag{3.18}$$

where u_{con} and v_{con} are the contravariant velocity components in the computational plane and are calculated from the relationships

$$u_{con} = \xi_x u + \xi_y v \tag{3.19}$$

$$v_{con} = \eta_x u + \eta_y v \tag{3.20}.$$

The contravariant velocities and speed of sound are calculated at each interior point of the initialized flow-field. The same time step is used throughout the entire flow-field for a constant CFL. Using the maximum characteristic velocity helps ensure stability at each point throughout the flow-field but may restrict convergence to a steady-state solution.

5. Stability Analysis

The elements of a Fourier or von Neumann stability analysis are presented for the two-step Runge-Kutta scheme.

For the Fourier or von Neumann stability analysis, the numerical solution of a finite-difference equation, N , is written

$$N = D + \beta \tag{3.21}$$

where D is the exact solution, and β is the round-off error which is machine dependent. A Fourier solution for a typical finite-difference equation is assumed to have round-off error in the form

$$\beta(x,t) = \sum_m e^{rt} e^{ik_m x} \tag{3.22}$$

where k_m is a Fourier coefficient. Equations (3.21) and (3.22) are substituted into the finite-difference equation, and the value for the error amplification factor is solved.

The stability criteria are determined from limiting the amplification factor :

$$\left| e^{r\Delta t} \right| \leq 1.0 \tag{3.23}.$$

For a system of equations, the stability criteria are determined based upon the largest eigenvalue of the system of equations in matrix form.

Equations (3.2) and (3.3) of the two-step Runge-Kutta scheme are repeated in the form :

$$\mathbf{Q}^* = \mathbf{Q}^n - \alpha \Delta t \left(\mathbf{A} \frac{\partial \mathbf{Q}^n}{\partial x} + \mathbf{B} \frac{\partial \mathbf{Q}^n}{\partial y} \right) \tag{3.24}$$

$$\mathbf{Q}^{n+1} = \mathbf{Q}^n - \Delta t \left(\mathbf{A} \frac{\partial \mathbf{Q}^*}{\partial x} + \mathbf{B} \frac{\partial \mathbf{Q}^*}{\partial y} \right) \tag{3.25}.$$

Central-differencing is used to calculate the flux terms, and the finite-difference equation in vector form for equation (3.24) is written

$$\mathbf{Q}^* = \mathbf{Q}^n - \alpha \Delta t \left(\mathbf{A} \frac{\mathbf{Q}^*_{x+\Delta x} - \mathbf{Q}^*_{x-\Delta x}}{2\Delta x} + \mathbf{B} \frac{\mathbf{Q}^*_{y+\Delta y} - \mathbf{Q}^*_{y-\Delta y}}{2\Delta y} \right) \qquad (3.26).$$

The amplification factor $\mathbf{G}^*$, where

$$\mathbf{Q}^* = \mathbf{G}^* \mathbf{Q}^n \qquad (3.27)$$

is derived to be

$$\mathbf{G}^* = \mathbf{I} - i\alpha \left(\frac{\Delta t}{\Delta x} \mathbf{SA} + \frac{\Delta t}{\Delta y} \mathbf{SB} \right) \qquad (3.28)$$

based upon the quasi-one-dimensional analysis found in Merkle[10]. **SA** and **SB** represent the sine of the components in the **A** and **B** matrices, and i is the square root of negative one. **I** is the identity matrix.

The finite-difference equation for the second step of the Runge-Kutta scheme, equation (3.25), is written

$$\mathbf{Q}^{n+1} = \mathbf{Q}^n - \Delta t \left(\mathbf{A} \frac{\mathbf{Q}^*_{x+\Delta x} - \mathbf{Q}^*_{x-\Delta x}}{2\Delta x} + \mathbf{B} \frac{\mathbf{Q}^*_{y+\Delta y} - \mathbf{Q}^*_{y-\Delta y}}{2\Delta y} \right) \qquad (3.29).$$

The total amplification factor **G**, where

$$Q^{n+1} = G\,Q^{n} \tag{3.30}$$

is evaluated to be

$$G = I - i\left(\frac{\Delta t}{\Delta x}SA + \frac{\Delta t}{\Delta y}SB\right)G^{*} \tag{3.31}.$$

Equations (3.28), (3.30) and (3.31) show that the stability of the two-step Runge-Kutta scheme depends upon the maximum CFL and the scalar α used. Chima [12] explains the maximum CFL of the two-step Runge-Kutta scheme is 1.0 to ensure a stable convergence of the flow-field solution. A CFL value of 0.81 was utilized with the two-step Runge-Kutta scheme to provide for some stability margin and resulted in the fastest convergence rate for the test cases attempted.

Furthermore, Chima [12] adds that the scalar α must be in the range of $0.5 \le \alpha \le 0.85$ for stability and that $\alpha \approx 0.6$ provides for the best convergence. Merkle [10] suggests using a scalar value of $\alpha = 0.615$, which was verified to provide faster convergence to a steady-state flow-field solution than the Taylor series derived value of $\alpha = 0.5$. Using the scalar value of $\alpha = 0.615$ makes the two-step Runge-Kutta scheme first-order accurate in time.

6. Dissipation Computation

The wave-like nature of the Euler equations requires the addition of numerical or artificial dissipation in order in reach a steady-state flow-field solution in a finite domain. Artificial viscosity is calculated and added to decay the transient solution of the Euler equations and for shock treatment in transonic flow. Second and fourth-order spatial derivatives are determined at each point in the flow-field.

These viscous-like terms are multipied by factors which are sufficiently small not to appreciably change the steady-state solution.

Dissipation terms are calculated using the routine written by Ekaterinaris [11]. Second-order derivatives, $\partial^2 \mathbf{Q} / \partial x^2$ and $\partial^2 \mathbf{Q} / \partial y^2$, are evaluated using finite-difference equations and added for shock treatment to both steps of the two-step Runge-Kutta scheme . Second-order dissipation factors are equal to zero for subsonic flow evaluation. For transonic flow, the dissipation factor is selected higher in the x direction than the y direction because greater property changes occur across the shock. The second-order x and y direction dissipation factors, $w2_x$ and $w2_y$, are selected by the user from ranges of 0.25 to 0.50 and 0.10 to 0.20 respectively. Second-order dissipation, $\mathbf{D}_2$, is calculated in the form

$$\mathbf{D}_2 = w2_x\left(\frac{\partial^2 \mathbf{Q}}{\partial x^2}\right) + w2_y\left(\frac{\partial^2 \mathbf{Q}}{\partial y^2}\right) \qquad (3.32).$$

Fourth-order artificial viscosity is used to decay the transient solution and is also added to both steps of the Runge-Kutta scheme. Fourth-order spatial derivatives, $\partial^4 \mathbf{Q} / \partial x^4$ and $\partial^4 \mathbf{Q} / \partial y^4$, are calculated at each point based upon the estimated second-order derivatives. The derivatives are multiplied by factors, $w4_x$ and $w4_y$, which are selected by the user within the range of 0.03 and 0.05. Mathematically, the fourth-order dissipation vector, $\mathbf{D}_4$, is formulated

$$\mathbf{D}_4 = w4_x\left(\frac{\partial^4 \mathbf{Q}}{\partial x^4}\right) + w4_y\left(\frac{\partial^4 \mathbf{Q}}{\partial y^4}\right) \qquad (3.33).$$

7. Delta Form of the Two-Step Runge-Kutta Scheme

The two-step Runge-Kutta scheme is implemented in delta form. The intermediate flow-field $\mathbf{Q}^*$ is calculated

$$\mathbf{Q}^* = \mathbf{Q}^n + \Delta\mathbf{Q}^* \tag{3.34}$$

where $\Delta\mathbf{Q}^*$ is composed of flux and dissipation terms computed from the $\mathbf{Q}^n$ flow-field,

$$\Delta\mathbf{Q}^* = \alpha\left[-\Delta t\left(\frac{\partial\mathbf{E}}{\partial x} + \frac{\partial\mathbf{F}}{\partial y}\right) + \mathbf{D}_2 + \mathbf{D}_4\right]^n \tag{3.35}.$$

Next, the updated flow-field, $\mathbf{Q}^{n+1}$, is computed :

$$\mathbf{Q}^{n+1} = \mathbf{Q}^n + \Delta\mathbf{Q}^{n+1} \tag{3.36}$$

where $\Delta\mathbf{Q}^{n+1}$ is composed of flux and dissipation terms computed from the $\mathbf{Q}^*$ flow-field,

$$\Delta\mathbf{Q}^{n+1} = -\Delta t\left(\frac{\partial\mathbf{E}}{\partial x} + \frac{\partial\mathbf{F}}{\partial y}\right)^* + \mathbf{D}_2^* + \mathbf{D}_4^* \tag{3.37}.$$

8. Calculation of Flux Terms and Dissipation

The flux terms, $\partial\mathbf{E}/\partial x$ and $\partial\mathbf{F}/\partial y$, and dissipation terms, $\mathbf{D}_2$ and $\mathbf{D}_4$, for both stages of the two-step Runge-Kutta scheme are computed in the computational

plane and transformed using the Jacobian of transformation for evaluation in the physical plane.

9. Boundary Evaluations

Explicit boundary conditions are enforced for both the intermediate and updated stages of the two-step Runge-Kutta scheme to calculate the flow-field properties at the boundary grid points.

The boundaries are defined at the airfoil surface, the inlet and exit boundaries, and the wake of the airfoil. Boundary conditions to be applied include the inviscid requirement for flow tangency on the airfoil surface and freestream conditions in the farfield. The properties at the boundaries which cannot be calculated from the boundary conditions are extrapolated from the interior flow-field or the farfield.

Flow tangency is first required at the surface of the airfoil for inviscid flow. This is enforced by setting the velocity component normal to the surface of the airfoil to zero in the computational plane,

$$v_{con}\Big|_{\text{airfoil surface}} = 0 \qquad (3.38)$$

Since there are four 2-D Euler equations and one boundary condition at each boundary, three properties along the airfoil's surface must be derived from the equations of motion. The tangential velocity components, density, and energy are extracted from the interior grid points.

Flow-field properties extrapolated from interior points at the inlet and exit boundaries are dependent upon the nature of the eigenvalues of the Jacobian matrices. For subsonic flight, three eigenvalues of both the **A** and **B** matrices are

positive, and one is negative. At the boundary points with incoming flow, three properties are extrapolated from the farfield and one property is extrapolated from interior grid points. Total pressure, velocity, and entropy are assigned freestream values at the inlet boundaries, and static pressure is extrapolated from interior grid points.

At the exit boundaries, three properties are extrapolated from the interior grid, and one property is assigned freestream values. Static pressure is assigned its freestream value, and density, entropy, and velocity are extrapolated from the interior points.

Flow-field properties in the wake of the airfoil are determined by averaging the properties on both sides of the wake.

10. Force and Moment Calculations

The coefficient of pressure, C_p, where

$$C_p = \frac{P - P_\infty}{\frac{1}{2}\rho_\infty V_\infty^2} \qquad (3.39),$$

is calculated for each grid point on the surface of the airfoil and stored in a file. Next, the coefficient of lift, C_l,

$$C_l = \frac{l}{\frac{1}{2}\rho_\infty V_\infty^2 c} \qquad (3.40)$$

and coefficient of wave drag, C_{dw} ,

$$C_{dw} = \frac{d_w}{\frac{1}{2}\rho_\infty V_\infty^{\,2} c} \tag{3.41}$$

are calculated where l is the two-dimensional lift, d_w, is two-dimensional wave drag with no viscous contributions, and c is airfoil chord length.

11. Storage

After the final iteration is performed, necessary information including the **Q** vector at each point is stored in a file for a future restart of the solution and for graphical analysis.

12. Residuals

A residual is computed after each iteration to check the convergence of the flow-field solution. Stability analysis requires a means to check the convergence and convergence rate of the flow-field solution. Density residuals are computed to measure the change in the values of density throughout the flow-field after each iteration.

Every iteration the density is updated at each point (i,j) in the form

$$\rho_{i,j}^{n+1} = \rho_{i,j}^{n} + \Delta\rho^{n+1}{}_{i,j} \tag{3.42}.$$

The density residual, θ, is calculated as the summation of the absolute values of the density changes throughout the flow-field,

$$\theta = \sum_{i=1}^{imax} \sum_{j=1}^{jmax} \left| \Delta \rho_{i,j} \right| \qquad (3.43) .$$

When a typical solution converges, θ initially increases and then steadily decreases for remaining iterations.

13. Variations

Several variations of the two-step Runge-Kutta scheme were examined and evaluated. Local time-stepping requires each grid point be assigned its own time step for advancement of the flow-field solution during each iteration. A time step for each point in the flow-field was calculated based upon a constant CFL and the contravariant velocities at each point throughout the flow-field. Local time-stepping did not accelerate the convergence of the two-step Runge-Kutta scheme and was not implemented in the final version of RK2EULER.

Explicit residual smoothing was also attempted to increase the convergence rate of the solution. Here, the delta form of the **Q** vector at each point is weighted with the changes in the **Q** vector at adjacent points for both steps of the Runge-Kutta scheme. Several variations of explicit smoothing were attempted, and none resulted in improved convergence

Different procedures for the application of dissipation were investigated in order to decrease computer processing time. The calculation of second and fourth-order dissipation for both stages of the Runge-Kutta scheme require the majority of computer processing time. When dissipation was only calculated in the updated flow-field, the scheme was always unstable, even if it was added to both stages.

Similarly, if boundary conditions were not applied each step of the two-step scheme, the scheme was unstable and a steady-state solution would not be reached.

14. Flowchart

A flowchart of the two-step Runge-Kutta scheme Euler flow-solver, RK2EULER, is shown in Figure 3.4.

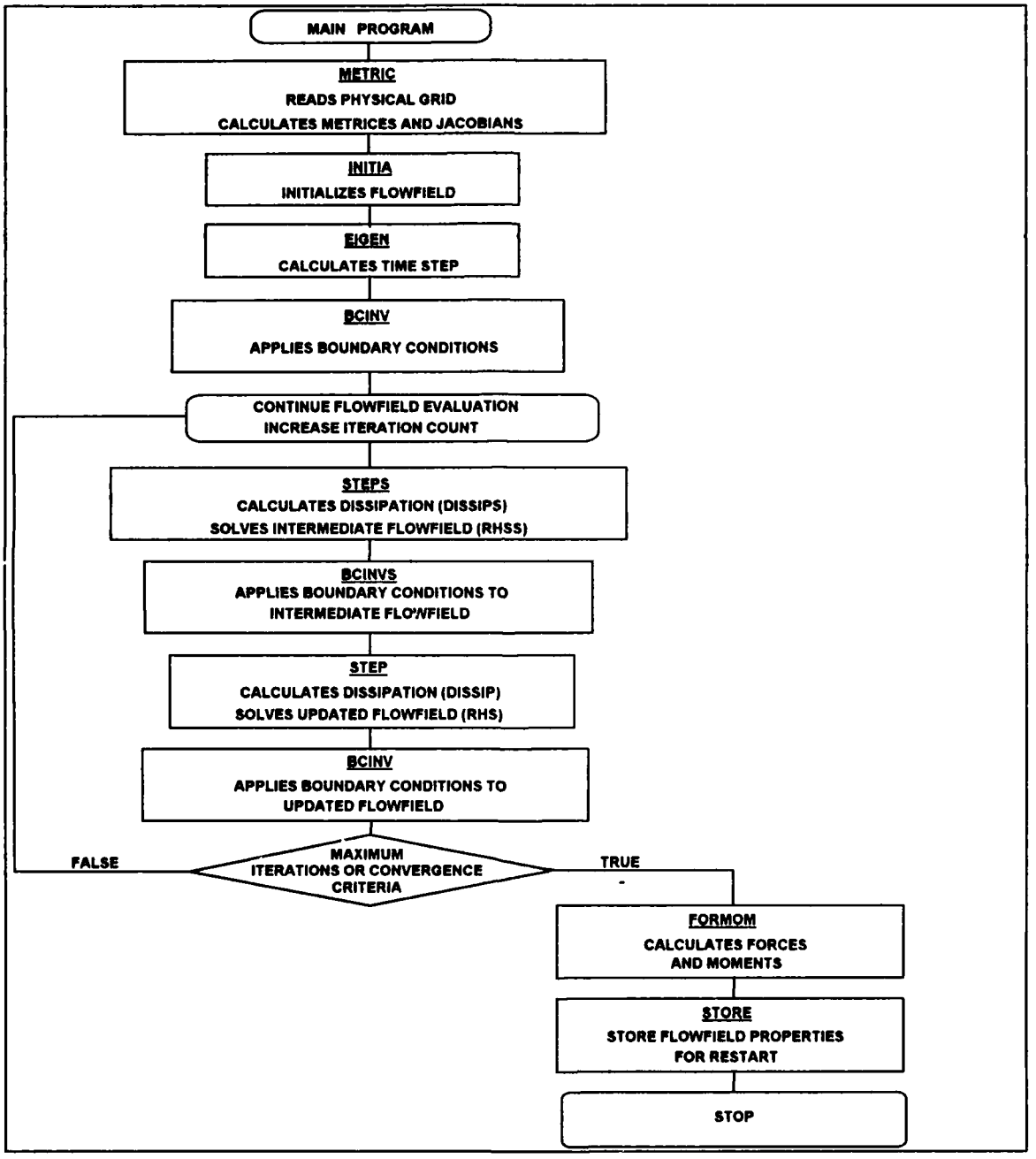

Figure 3.4 : Flowchart of RK2EULER

C. COMPARISON WITH A CRANK-NICHOLSON EULER SOLVER

1. Purpose

RK2EULER was compared with a similar Crank-Nicholson scheme Euler flow solver developed by J. A. Ekaterinaris [11]. The purpose of the comparison was to determine which flow solver could evaluate the pressure distribution around an airfoil in subsonic or transonic flow in the shortest amount of computer processing time. The fastest flow solver would then be coupled to an optimization routine to be utilized in the design of airfoils to match a target pressure distribution.

Three test cases were used for the comparison. These test cases are also utilized in the evaluation of the optimization routine. The stopping criterion for the flow solvers was chosen to determine when the steady-state pressure distribution around an airfoil was computed. Trial and error with both flow solvers in these test cases were used to determine when this criterion was satisfied.

Both flow solvers were programmed to end their flow-field evaluations when the summation of the square of the change in the coefficient of pressure after 200 iterations was less than 0.1% of the summation of the square of the C_p around the airfoil. Mathematically the stopping criterion is written as

$$\frac{\sum_{i=itel}^{iteu}\left(C_p^n - C_p^{n-200}\right)^2}{\sum_{i=itel}^{iteu}\left(C_p^n\right)^2} < 0.001 \qquad (3.38)$$

where $\sum_{i=itel}^{iteu}$ is the summation at the grid points on the airfoil surface from the lower trailing edge point to the upper trailing edge point and n is the iteration count.

All the grids used in the test cases were generated using the program GRAPE (Grids about Airfoils using Poisson's Equation) developed by Sorenson [13]. A Stardent workstation with vector processing capabilities was used for each test case, and both schemes were compiled to optimize vectorization.

2. Test Case 1

For the first test case, the inviscid flow-field around a NACA 0012 symmetric airfoil was calculated. The freestream Mach number was set at 0.6, and the airfoil was given an angle of attack of 0 degrees. A course 133 x 34 grid was generated for the analysis of the Euler equations and is shown in Figure 3.5.

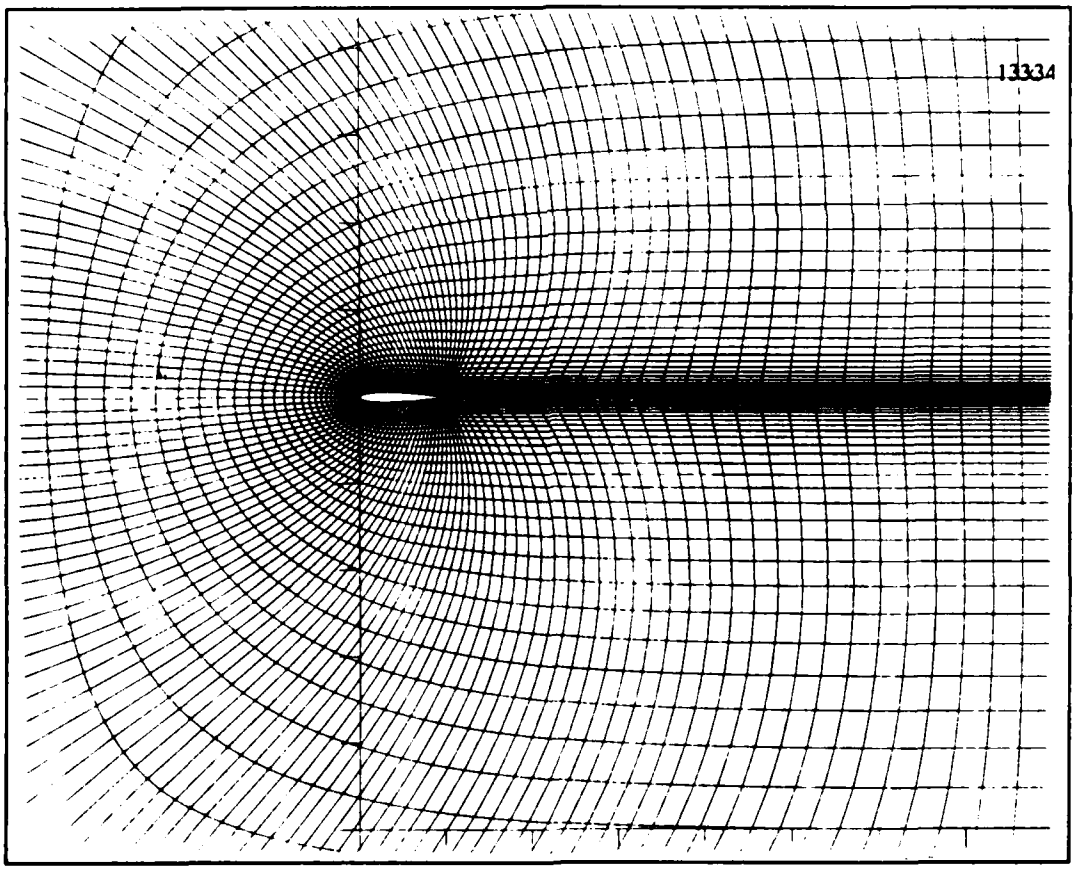

Figure 3.5 : 133 x 34 Grid around NACA 0012 airfoil

The CFL for the Crank-Nicholson scheme was set at 5.0, which is the optimum CFL for this scheme according to Merkle [10]. The two-step Runge-Kutta scheme utilized a CFL of 0.81 for the scalar $\alpha = 0.615$.

The Mach contours generated by the Crank-Nicholson and the Runge-Kutta schemes are given in Figure 3.6 and Figure 3.7 respectively. The two solutions are nearly identical. Slight differences exist away from the surface of the airfoil where the steady-state pressure distribution is calculated for the stopping criterion.

The convergence histories of the two flow solvers are illustrated in Figure 3.8, which shows the logarithm of the density residuals for each iteration count. The convergence rate for both solutions were similar with the two-step Runge-Kutta scheme having a higher initial residual.

The results of the first test case are shown in Table 3.1. The two-step Runge-Kutta scheme reached a flow-field solution based upon the stopping criterion 6.6 times faster than the Crank-Nicholson scheme, even through 200 more iterations were required for the Runge-Kutta scheme. This is primarily due to the vectorizability of the Runge-Kutta scheme which does not require matrix inversion.

Table 3.1 : Comparison of Flow Solvers
NACA 0012 133 x 34 grid
M = 0.6 AOA = 0.0

Flow Solver Scheme	CFL	Iterations	CPU time (min:sec)
Crank Nicholson Scheme	5.0	1000	34:01
Two-Step Runge-Kutta scheme	0.81	1200	5:08

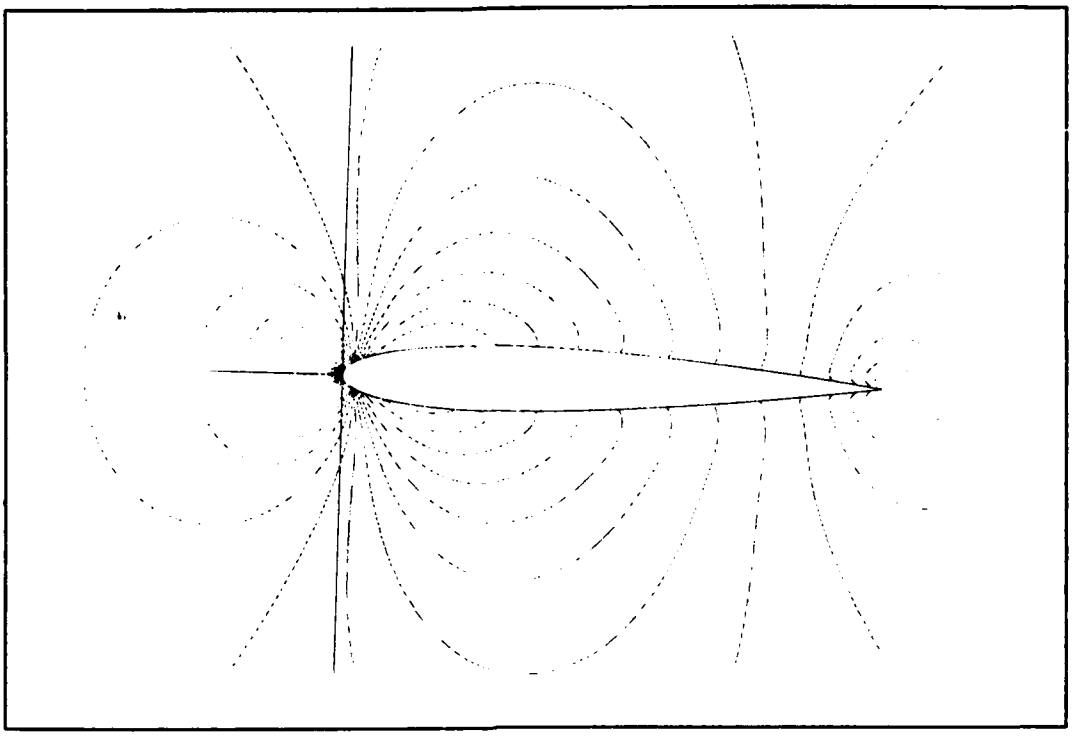

Figure 3.6 : Mach Contours from Crank-Nicholson Scheme

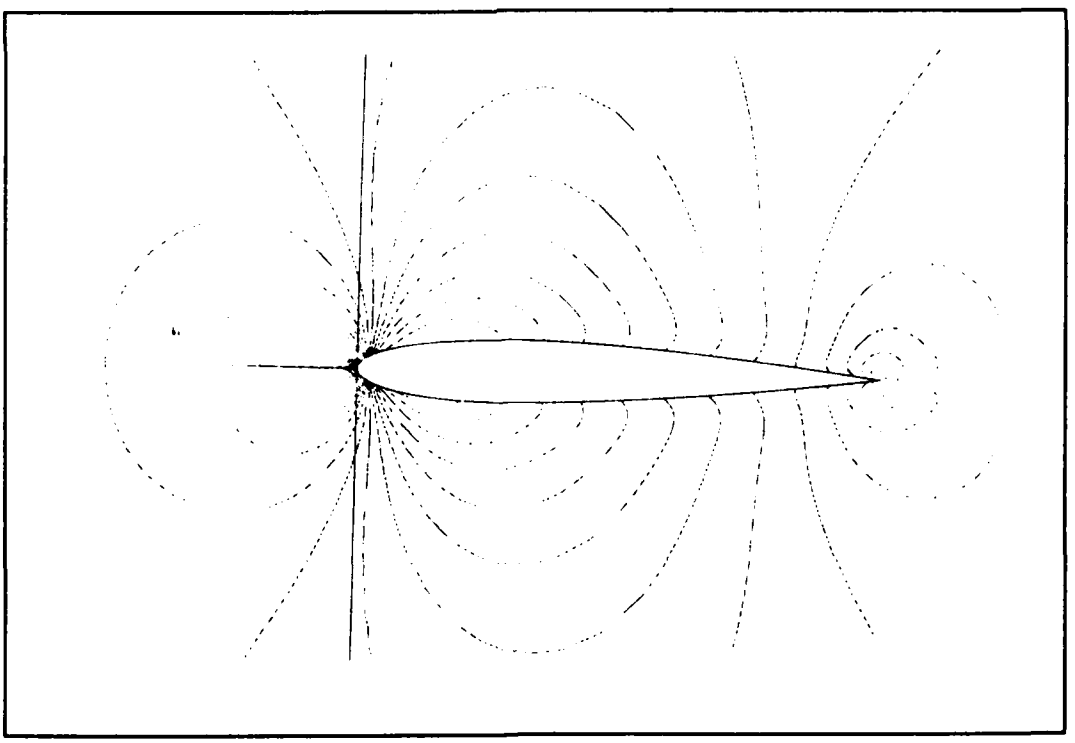

Figure 3.7 : Mach Contours from Runge-Kutta Scheme

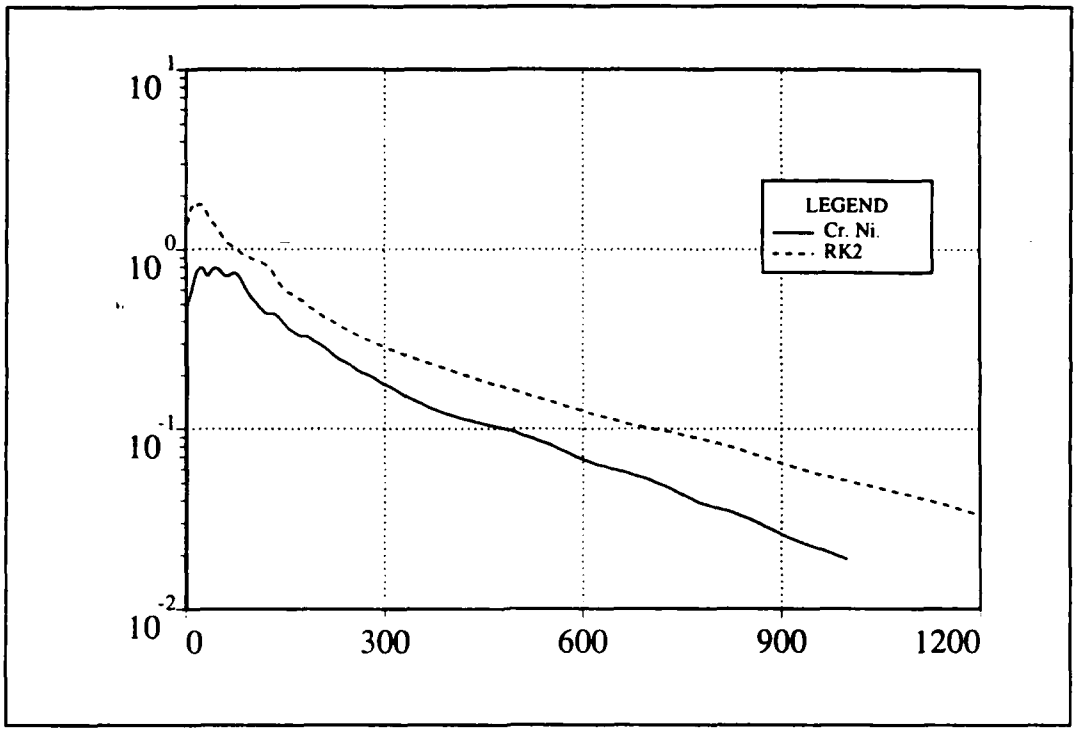

Figure 3.8 : Convergence History for Crank-Nicholson and Runge-Kutta Schemes

The same test case was run on a finer grid for comparison. The 201 x 52 grid shown in Figure 3.9 was used with both flow solvers. The larger number of grid points requires the inversions of larger matrices for the Crank-Nicholson scheme and approximately twice the number of point-by-point updates for the Runge-Kutta scheme.

The Mach contours generated by the Crank-Nicholson and the Runge-Kutta schemes are given in Figure 3.10 and Figure 3.11 respectively. Once again the solutions are similar with minor differences noticeable in the wakes of the airfoils.

The convergence histories for the two solutions are shown in Figure 3.12. The Runge-Kutta solution's convergence rate decreases as the number of iterations increases. The Crank-Nicholson solution's convergence rate oscillated around an average convergence rate which was higher than the Runge-Kutta's decreased rate.

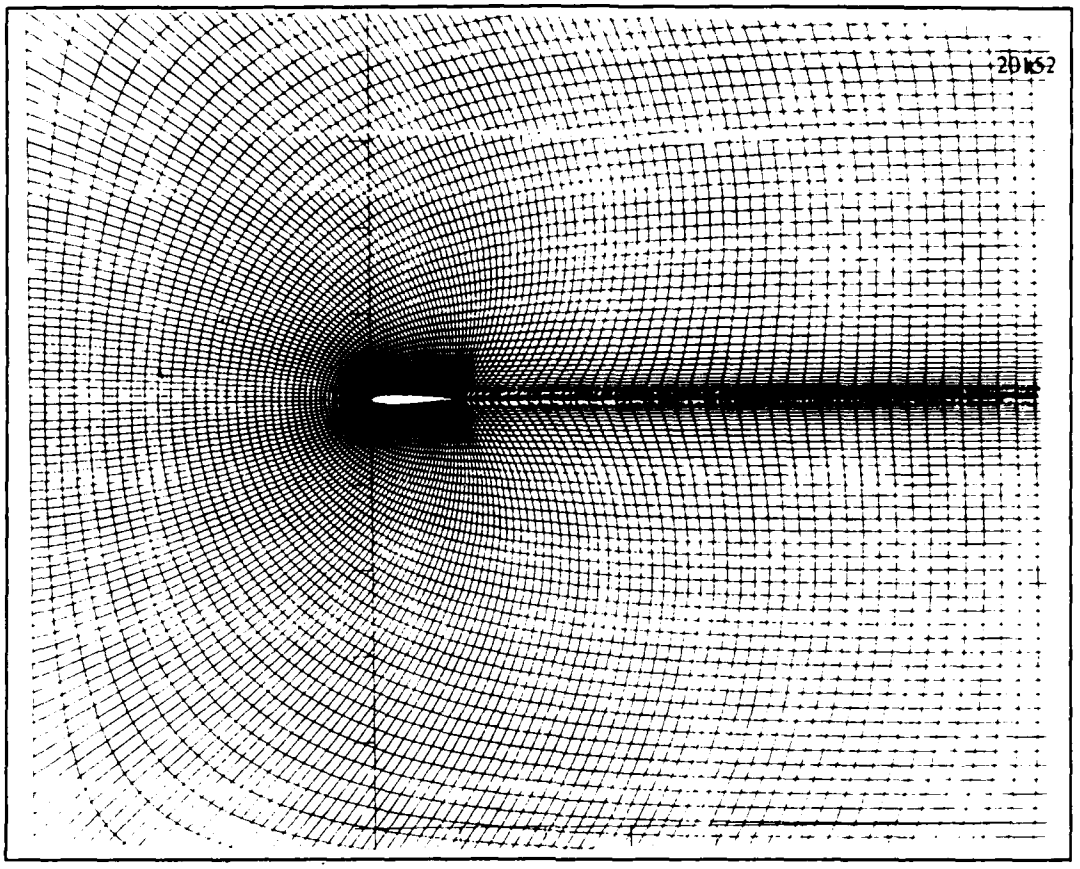

Figure 3.9 : 201 x 52 Grid around NACA 0012

The results are shown in Table 3.2. The two-step Runge-Kutta scheme reached a flow-field solution over 5 times faster than the Crank-Nicholson scheme. The Crank-Nicholson flow solver required fewer iterations to solve the steady-state pressure distribution using the greater number of grid points. Also, the Mach contours of the flow-field solutions were similar when both grid sizes were used with the Crank-Nicholson and Runge-Kutta schemes.

The Runge-Kutta solution's convergence rate decreased as the iteration count increased with the larger number of grid points, which was not apparent with the smaller number of grid points. This behavior of the convergence history is directly related to the low CFL of 0.81 utilized for the two-step Runge-Kutta scheme. The

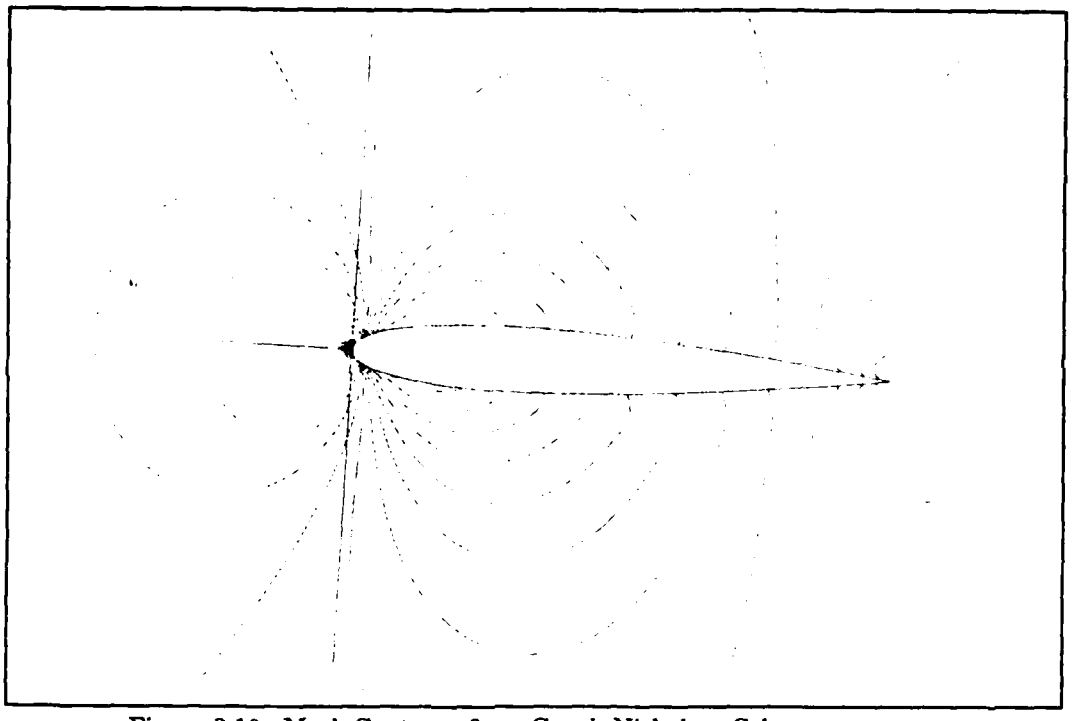

Figure 3.10 : Mach Contours from Crank-Nicholson Scheme

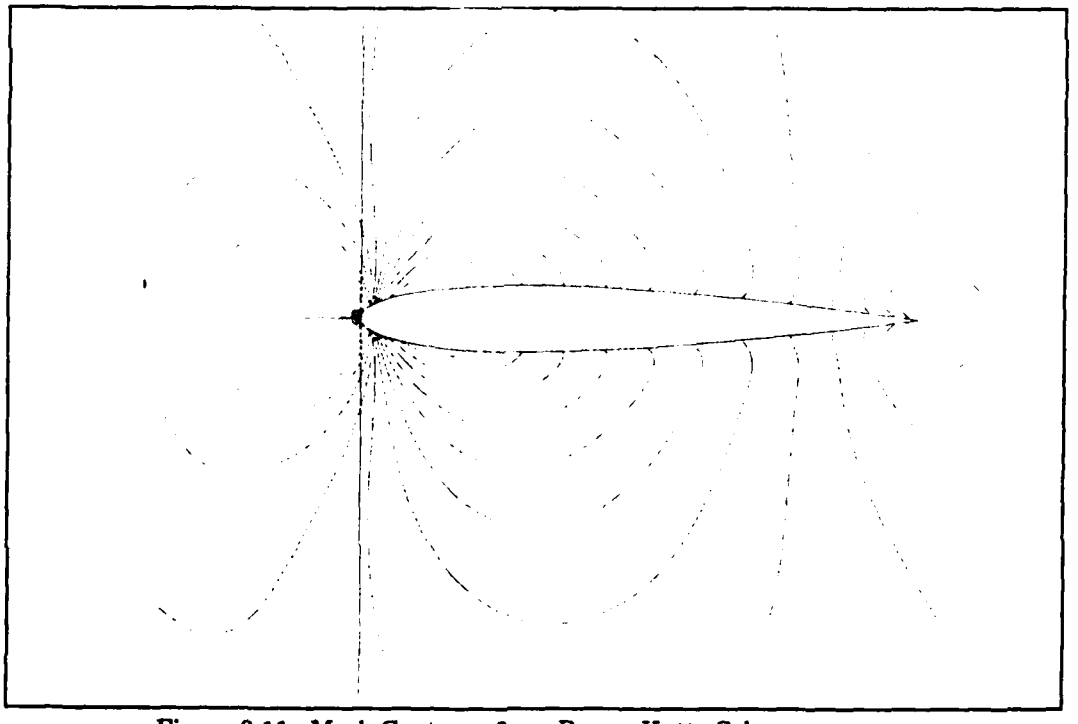

Figure 3.11 : Mach Contours from Runge-Kutta Scheme

convergence rate was constant when using both grid sizes for the Crank-Nicholson flow solver with a CFL of 5.0. The Crank-Nicholson flow solver was run at a CFL of 0.81 and its convergence history is shown in Figure 3.13. The convergence of the Crank-Nicholson solution with the smaller CFL is similar to the Runge-Kutta solution with a decreased convergence rate requiring 1200 iterations to reach the stopping criterion.

Table 3.2 : Comparison of Flow Solvers
NACA 0012 201 x 52 grid
M = 0.6 AOA = 0.0

Flow Solver Scheme	CFL	Iterations	CPU time (min:sec)
Crank Nicholson Scheme	5.0	800	63:39
Two-Step Runge-Kutta scheme	0.81	1200	10:29

3. Test Case 2

The second test case compares the flow solvers for subsonic flow over a NACA 2412 airfoil for a freestream Mach number of 0.6 with 2 degrees angle of attack. The 133 x 34 grid is shown in Figure 3.14.

The Mach contours generated by the Crank-Nicholson and the Runge Kutta schemes are given in Figure 3.15 and Figure 3.16 respectively. The two solutions are also nearly identical for the same stopping criteria.

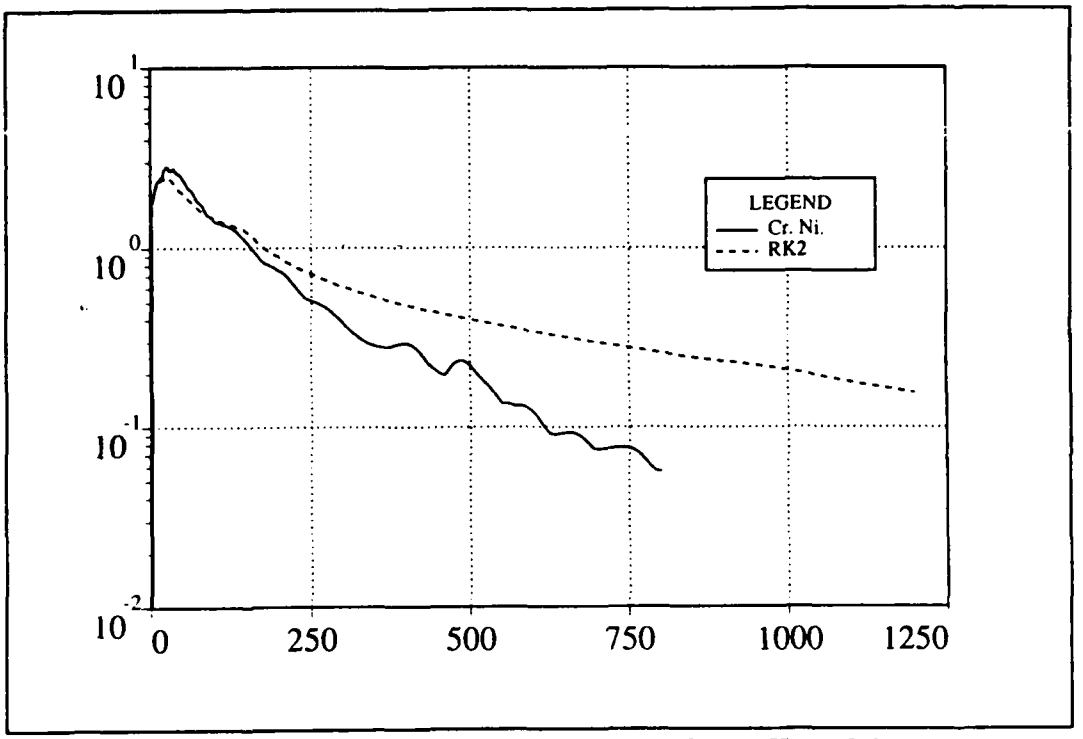

Figure 3.12 : Convergence History for Crank-Nicholson and Runge-Kutta Schemes

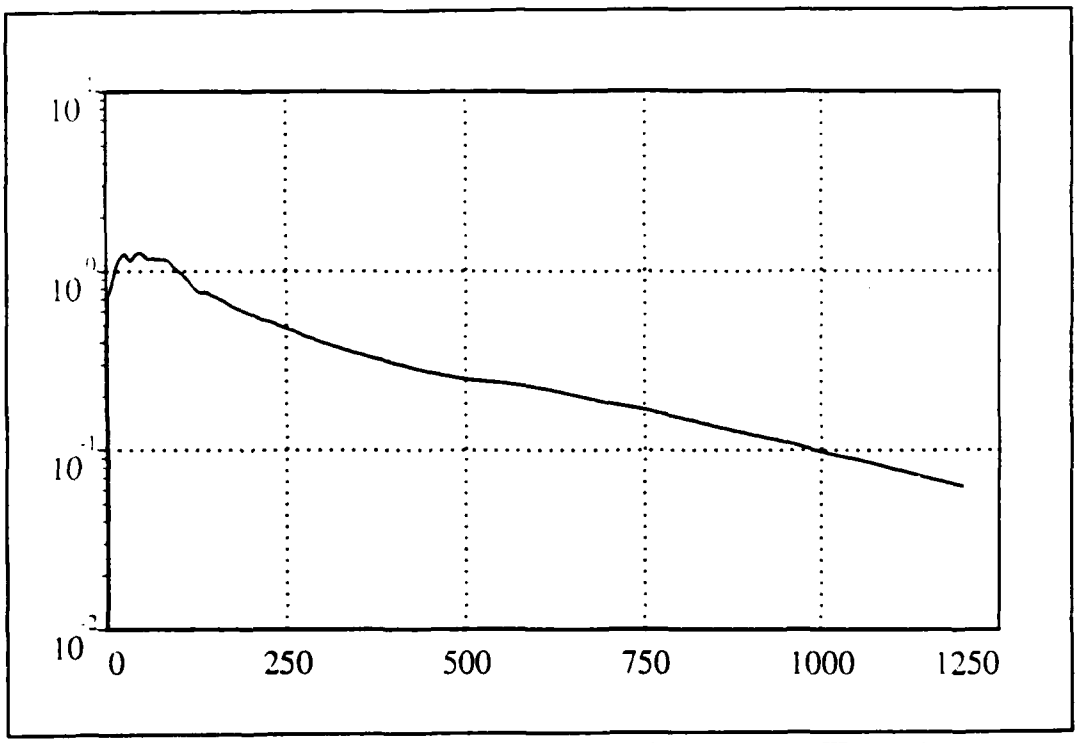

Figure 3.13 : Convergence History for Crank-Nicholson Scheme with CFL = 0.81

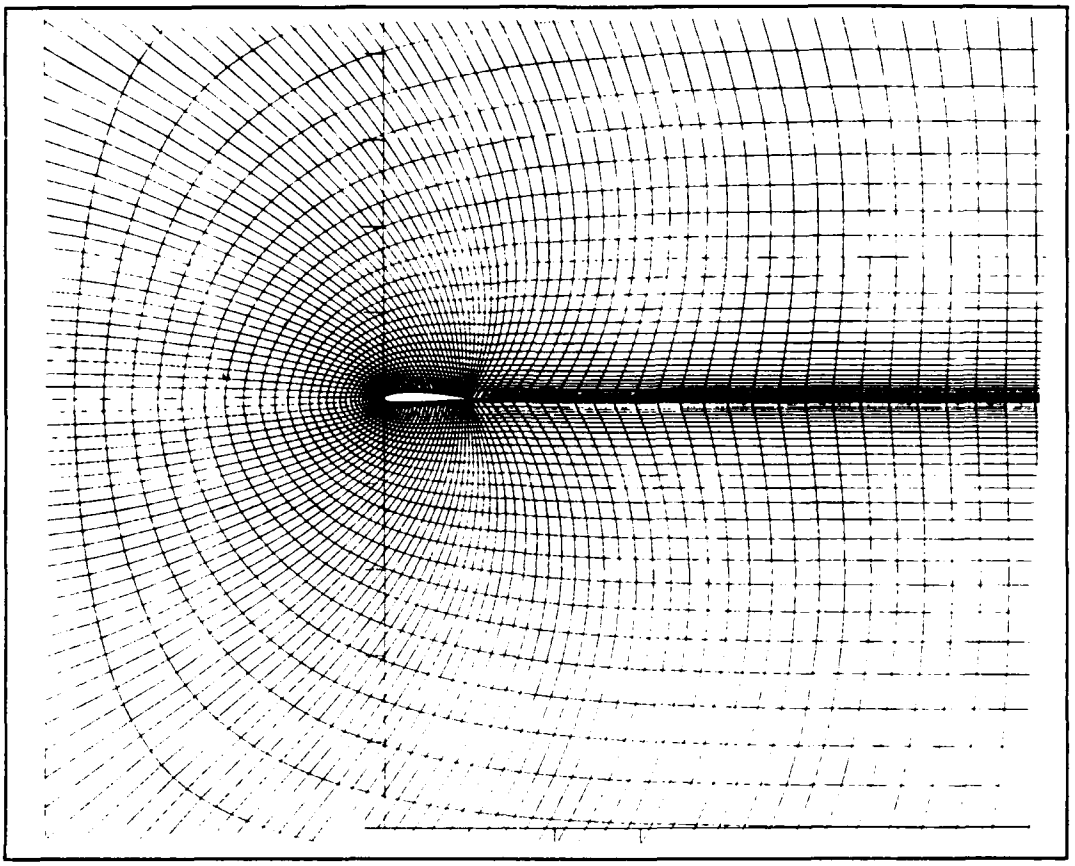

Figure 3.14 : 133 x 34 Grid around NACA 2412

The convergence histories for both solutions are shown in Figure 3.17. Again the convergence rate for the Runge-Kutta scheme decreased after higher numbers of iterations. After approximately 500 iterations, the convergence rate of the Crank-Nicholson flow solver decreases.

The results are shown in Table 3.3. The two-step Runge-Kutta scheme reached a flow-field solution roughly 5 times faster than the Crank-Nicholson scheme. More iterations were required to reach a solution for both flow solvers than in the first test case because of the more complicated flow-field around the cambered airfoil.

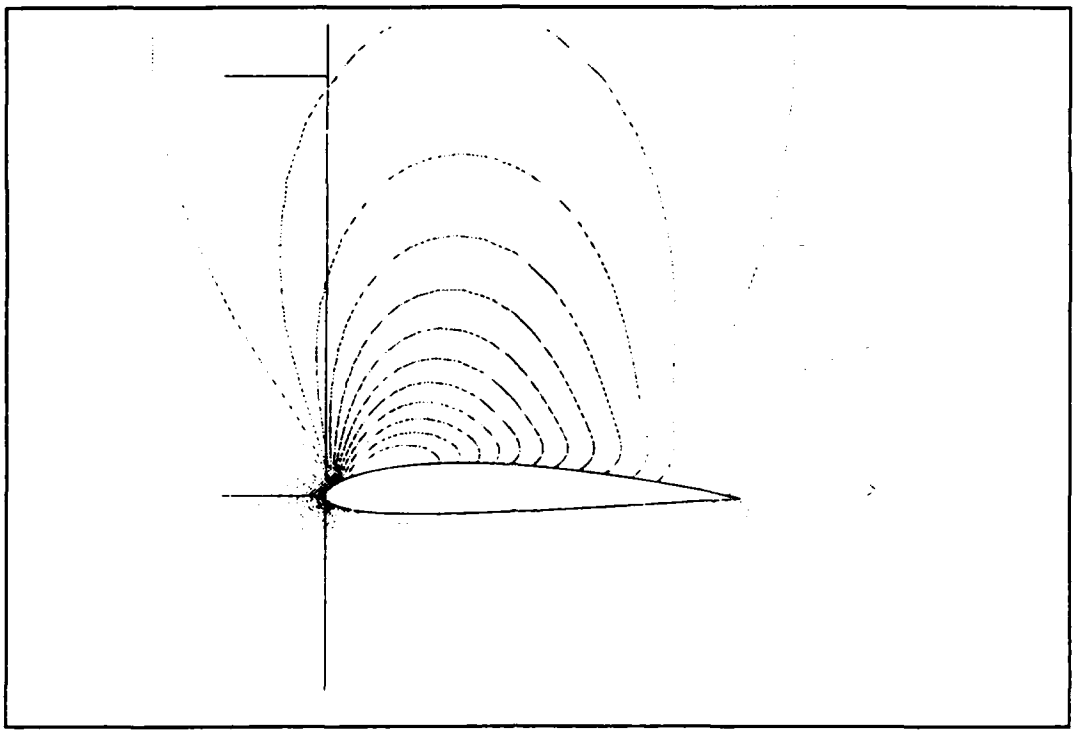

Figure 3.15 : Mach Contours from Crank-Nicholson Scheme

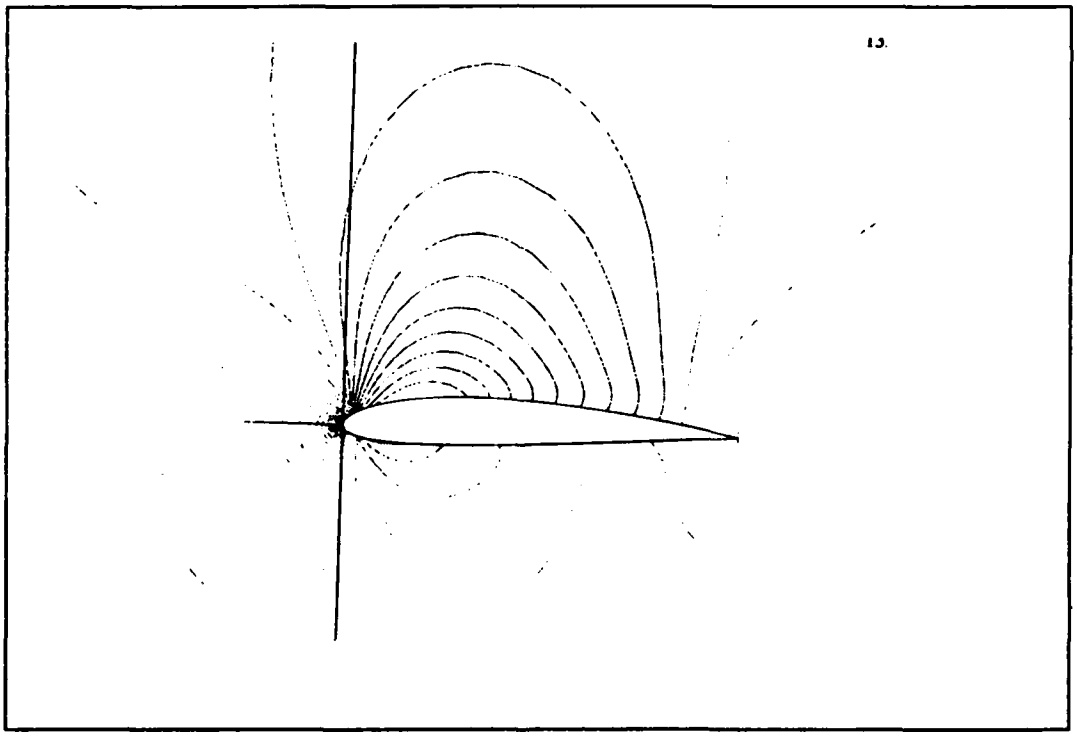

Figure 3.16 : Mach Contours from Runge-Kutta Scheme

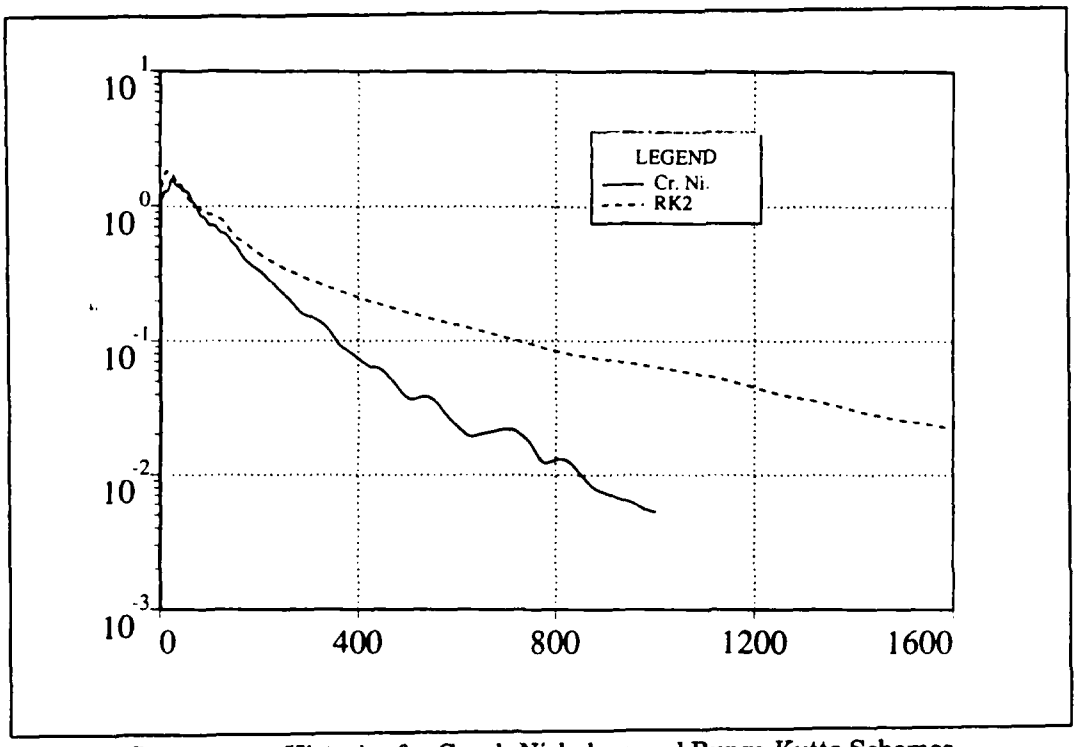

Figure 3.17 : Convergence Histories for Crank-Nicholson and Runge-Kutta Schemes

Table 3.3 : Comparision of Flow Solvers
NACA 2412 133 x 34 grid
M = 0.6 AOA = 2.0

Flow Solver Scheme	CFL	Iterations	CPU time (min:sec)
Crank Nicholson Scheme	5.0	1000	36:34
Two-Step Runge-Kutta Scheme	0.81	1600	7:25

The same test case was run using a finer grid with 201 x 52 grid points shown in Figure 3.18 to compare with results from the coarser grid. The Mach contours

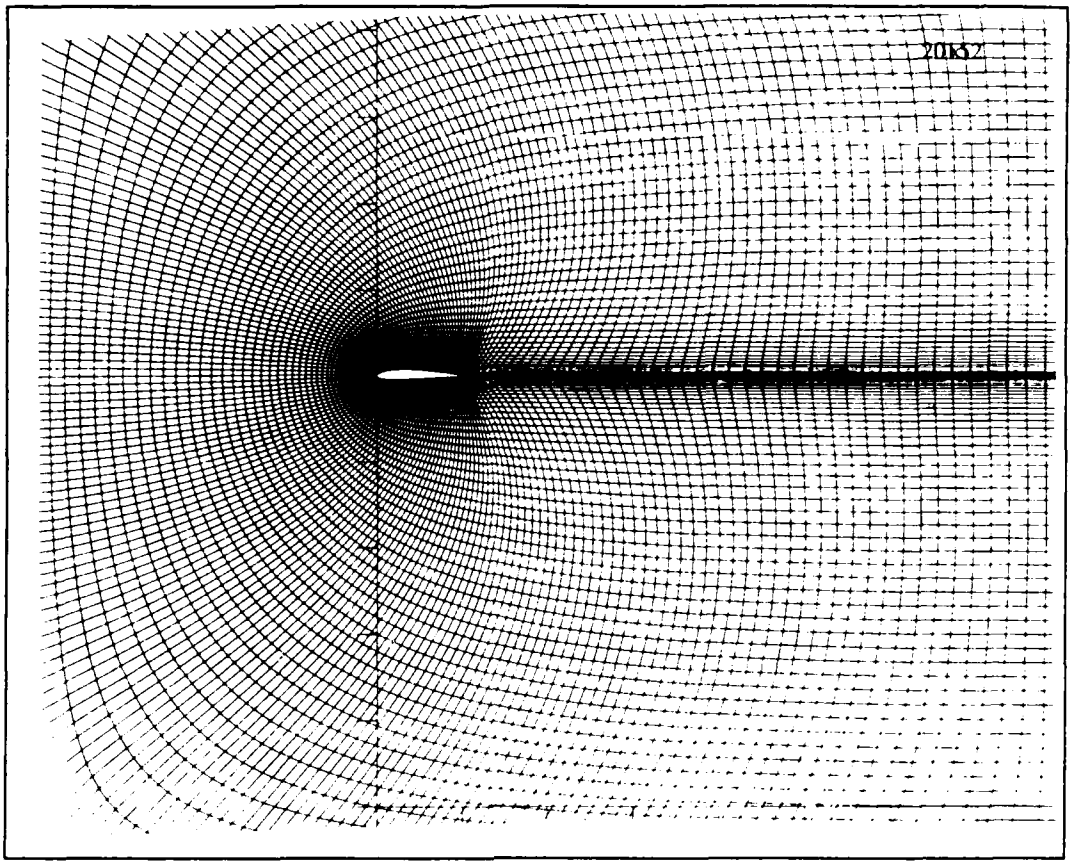

Figure 3.18 : 201 x 52 Grid around NACA 2412

generated from solutions using the Crank-Nicholson and the two-step Runge-Kutta flow solvers are shown in Figures 3.19 and 3.20 respectively. The solutions are again similar using both schemes.

The convergence histories of both solutions are show in Figure 3.21. The Crank-Nicholson flow solver converges faster per iteration than the Runge-Kutta scheme, however the convergence rate of both schemes decreases at higher iteration cycles.

A comparison of the performance of both flow solvers for the evaluation of the subsonic flow-field over a NACA 2412 airfoil at 2 degrees angle of attack with a

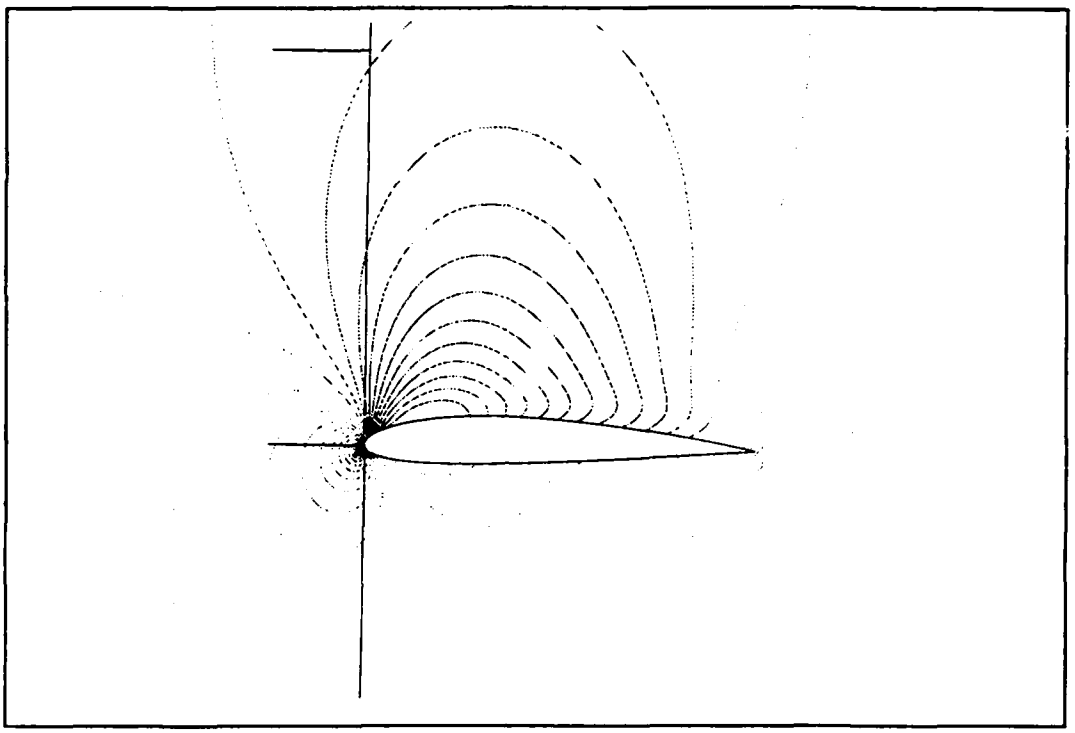

Figure 3.19 : Mach Contours from Crank-Nicholson Scheme

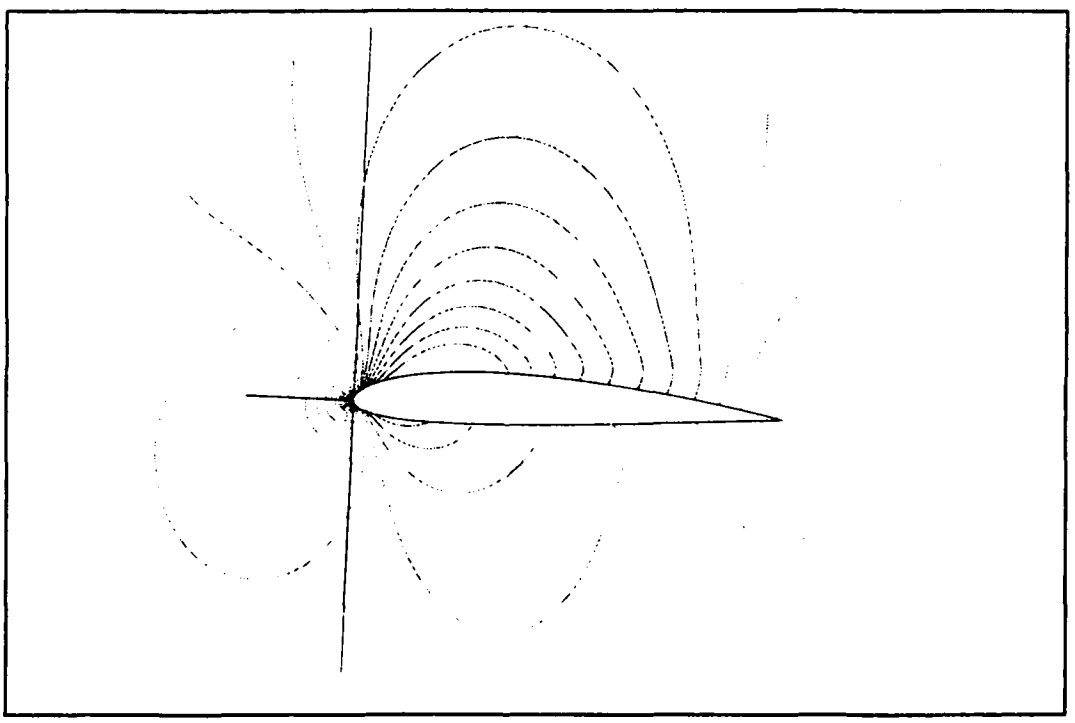

Figure 3.20 : Mach Contours from Runge-Kutta Scheme

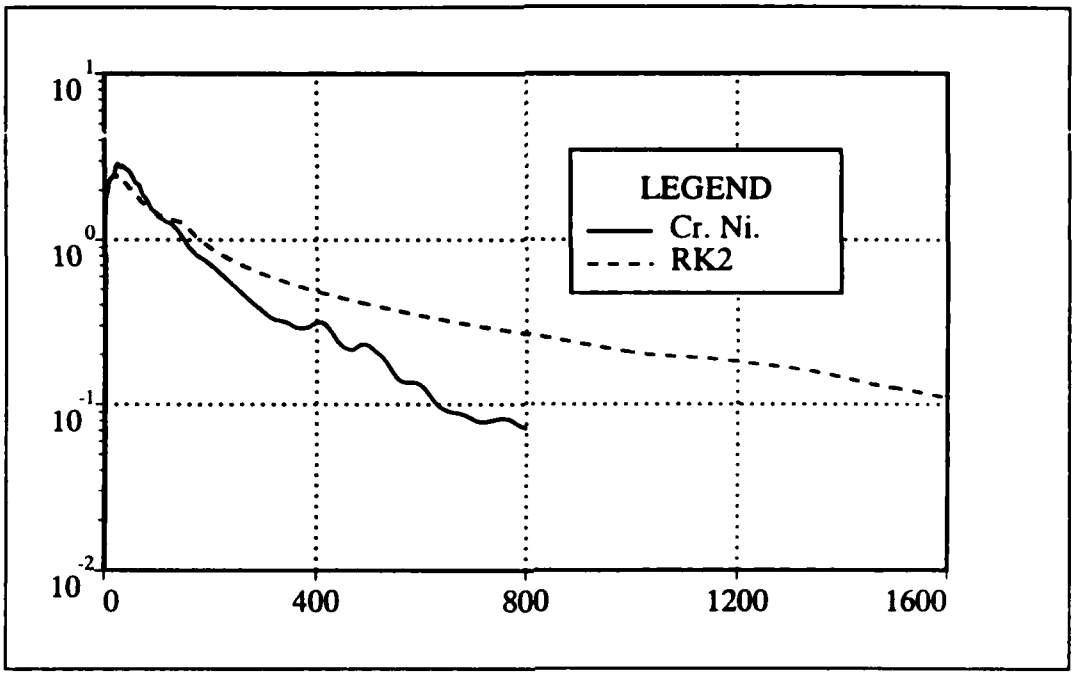

Figure 3.21: Convergence Histories for Crank-Nicholson and Runge-Kutta Schemes

201 x 52 grid is shown in Table 3.4. The Crank-Nicholson flow solver reached a solution in half the number of iterations that the Runge-Kutta flow solver required. However, the Runge-Kutta flow solver reached a solution 4.5 times faster than the Crank-Nicholson one because of the vectorizability of the explicit scheme.

Table 3.4 : Comparison of Flow Solvers
NACA 2412 201 x 52 grid
M = 0.6 AOA = 2.0

Flow Solver Scheme	CFL	Iterations	CPU time (min:sec)
Crank-Nicholson Scheme	5.0	800	64:57
Two-Step Runge-Kutta Scheme	0.81	1600	14:28

4. Test Case 3

The third test case compares the flow solvers ability to reach the inviscid solution for an airfoil generating lift in transonic flight. A NACA 0012 airfoil at a freestream Mach number of 0.8 and 0.5 degrees angle of attack is evaluated using the Euler flow solvers. The grids used are the same as those in Test Case 1.

The Mach contours generated by the Crank-Nicholson and the two-step Runge-Kutta flow solvers using a 133 x 34 grid are given in Figure 3.22 and Figure 3.23 respectively. The two solutions are also similar for the same stopping criterion. The solutions show the approximate locations of shocks on the upper and lower surfaces of the airfoils for inviscid flow. The lower shock is better defined by the two-step Runge-Kutta scheme; this illustrates that second-order dissipation added for shock treatment was more effective in this case with the Runge-Kutta scheme.

The convergence history of the density residuals are plotted against the iteration count for both solutions in Figure 3.24. The Crank-Nicholson and Runge-Kutta flow solvers both require more iterations than in the previous test cases. Also, the convergence rate for both solutions decreases after roughly 200 iterations.

The results are shown in Table 3.5. The two-step Runge-Kutta scheme calculated a steady-state pressure distribution around the airfoil 5.4 times faster than the Crank-Nicholson scheme.

The Crank-Nicholson and Runge-Kutta flow solvers were also used to calculate the flow-field solution around a NACA 0012 airfoil using a 201 x 52 grid and the same freestream conditions. The finer grid allows a more accurate prediction of the position of the shocks along the airfoil. The Mach contours generated from both solutions are shown in Figure 3.25 and Figure 3.26. Again the Runge-Kutta scheme provided a better resolution of the lower shock position.

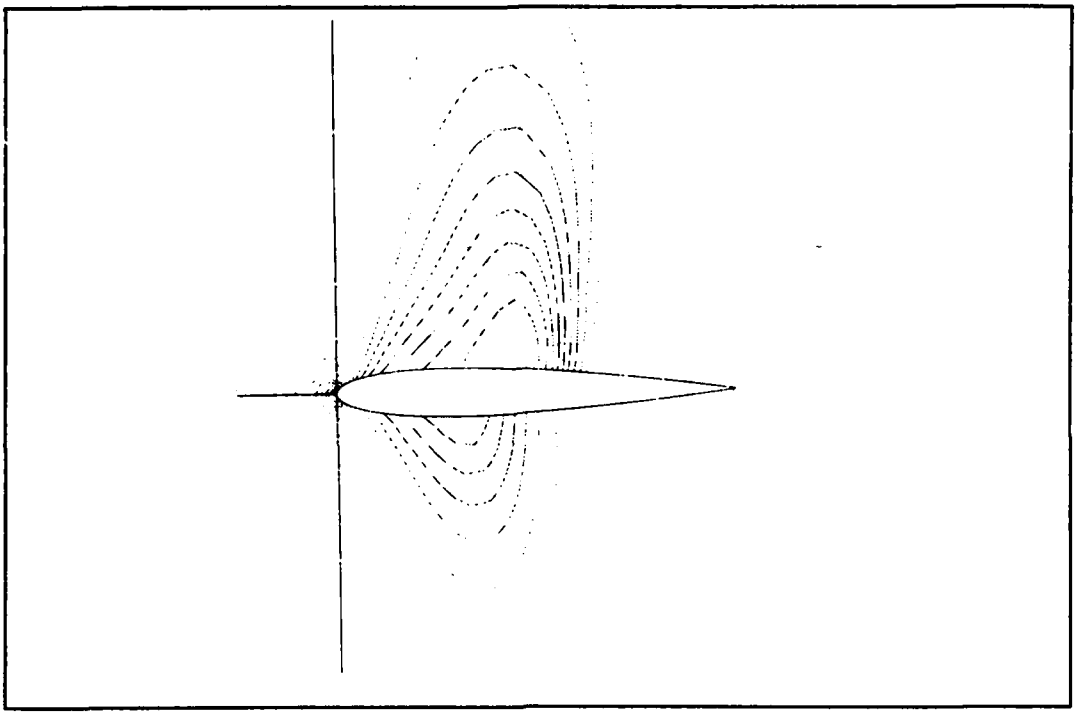

Figure 3.22 : Mach Contours from the Crank-Nicholson Scheme

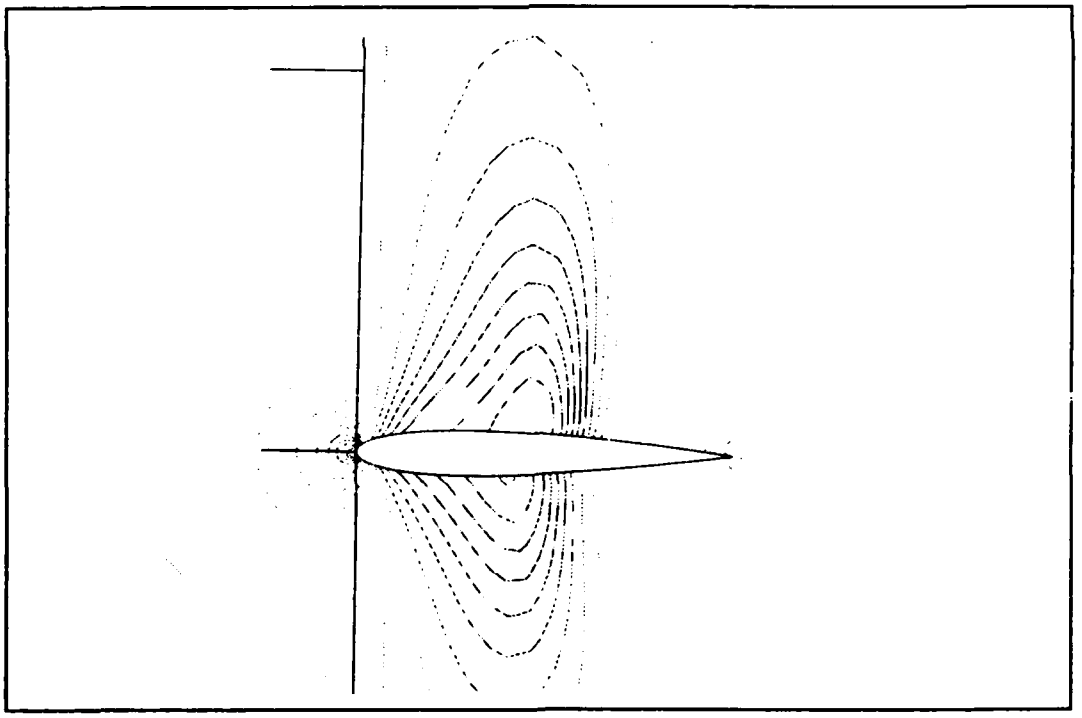

Figure 3.23 : Mach Contours from Runge-Kutta Scheme

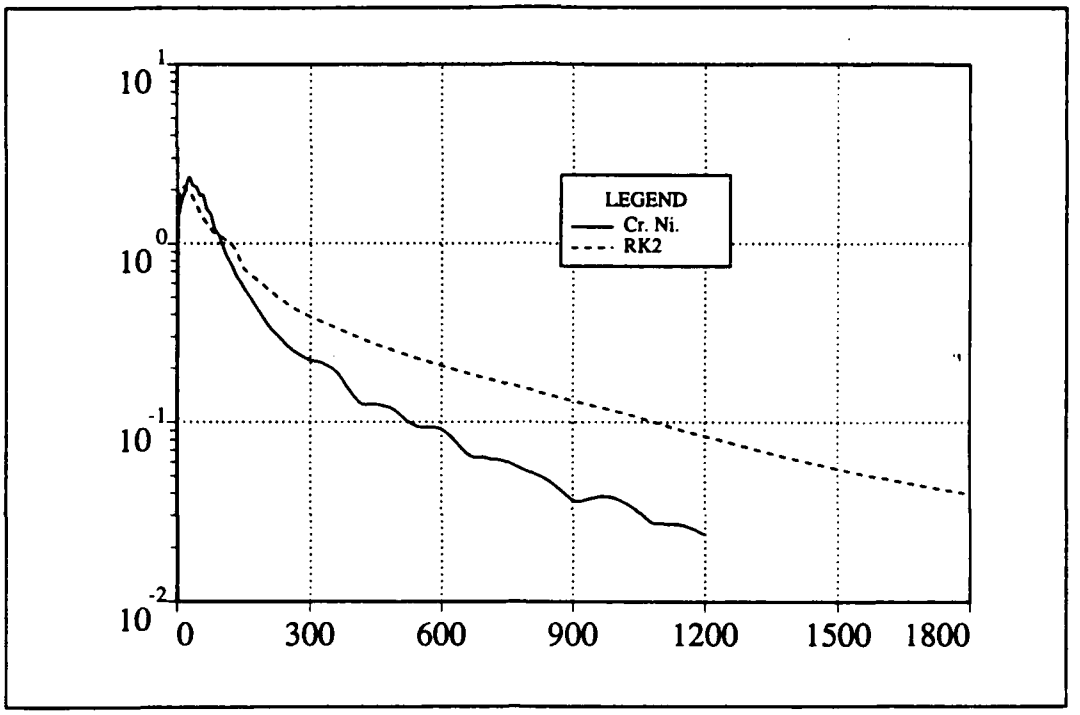

Figure 3.24 : Convergence Histories of Crank-Nicholson and Runge-Kutta Schemes

Table 3.5 : Comparision of Flow Solvers
NACA 0012 133 x 34 grid
M = 0.8 AOA = 0.5

Flow Solver Scheme	CFL	Iterations	CPU time (min:sec)
Crank Nicholson Scheme	5	1200	41:52
2 Stage Runge Kutta scheme	0.81	1800	7:43

The convergence histories of both flow solvers for the transonic test case using the 201 x 52 grid are shown in Figure 3.27. For this case, the Crank-Nicholson flow solver reached a higher initial residual and showed a similar decrease in

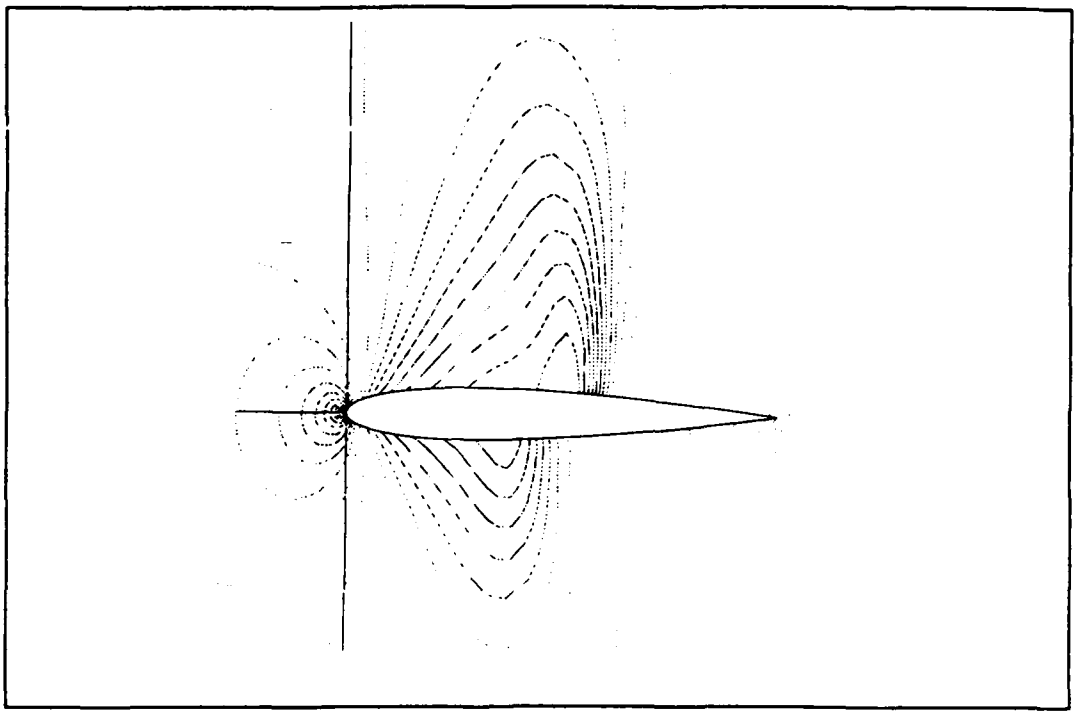

Figure 3.25 : Mach Contours from Crank-Nicholson Scheme

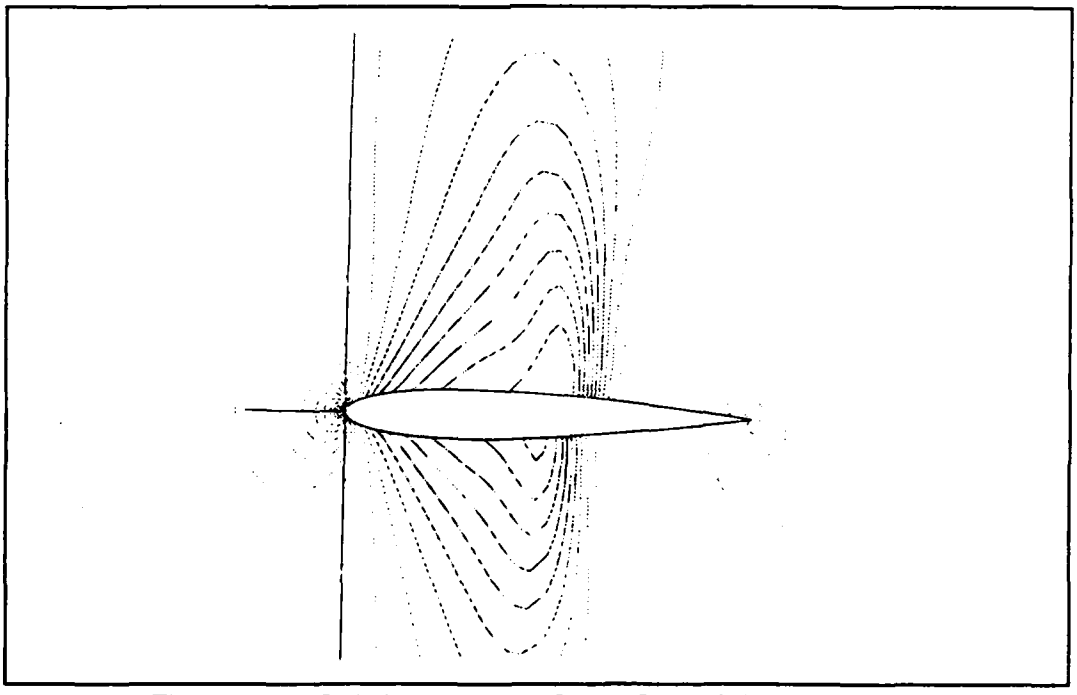

Figure 3.26 : Mach Contours from Runge-Kutta Scheme

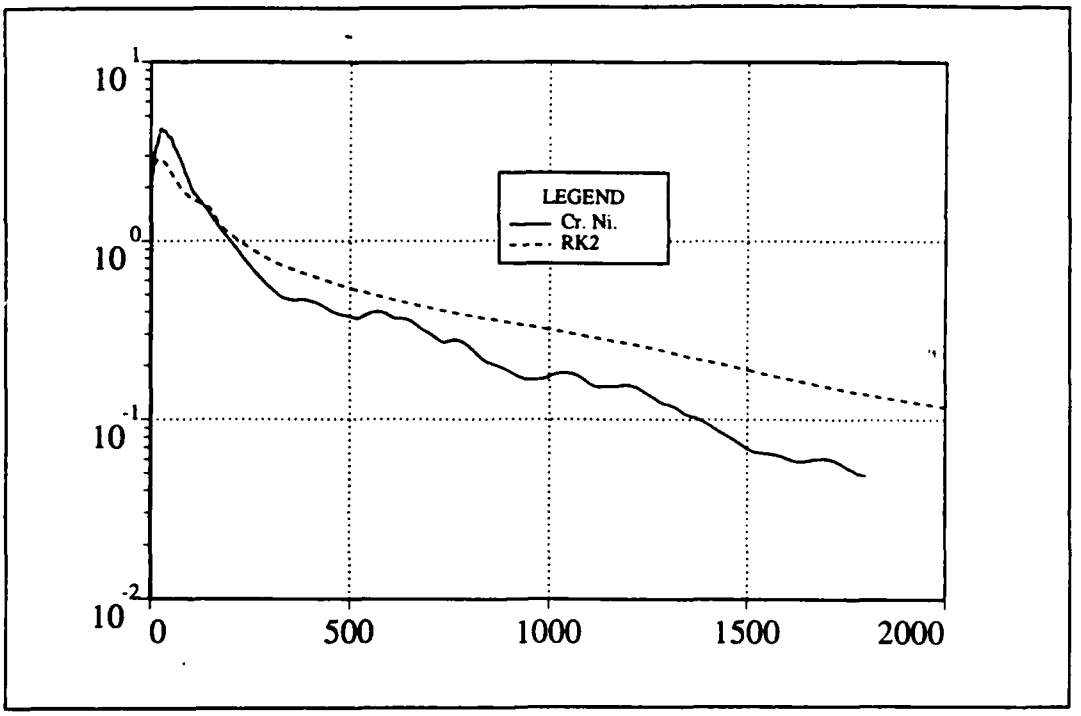

Figure 3.27 : Convergence Histories of Crank-Nicholson and Runge-Kutta Schemes

the convergence rate to the Runge-Kutta solution. Unlike the other test cases, the Crank-Nicholson flow solver took 50 % more iterations to reach a solution than with the coarser grid. These factors gave the Runge-Kutta solution a more impressive speed advantage than in the previous test cases.

A comparison of both flow solvers are shown in Table 3.6. The two-step Runge-Kutta scheme reached a flow-field solution over 8 times faster than the Crank-Nicholson scheme.

The Crank-Nicholson flow solver was run for this final test case with the fine grid using a CFL of 0.81. Its convergence history is shown in Figure 3.28. The Crank-Nicholson scheme showed neutral stability characteristics at higher iterations and required more iterations to reach a solution. This demonstrates a larger CFL can provide more robustness and a faster convergence of a solution.

Table 3.6 : Comparision of Flow Solvers
NACA 0012 201 x 52 grid
M = 0.8 AOA = 0.5

Flow Solver Scheme	CFL	Iterations	CPU time (min:sec)
Crank-Nicholson Scheme	5.0	1800	146:56
Two-Step Runge-Kutta Scheme	0.81	2000	18:04

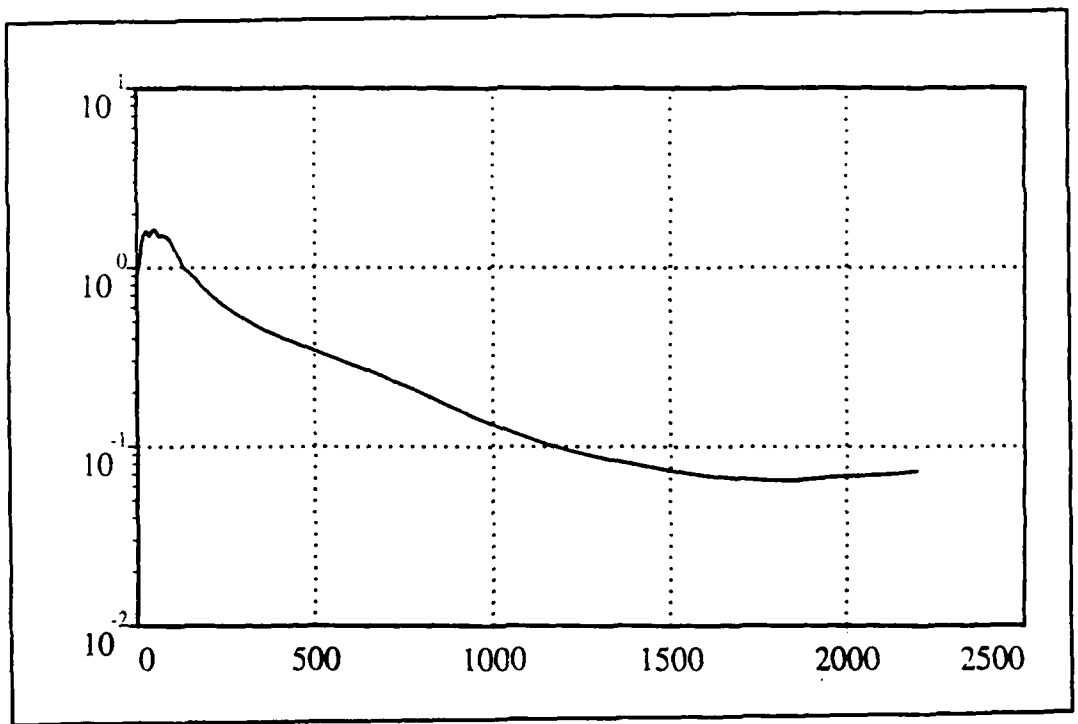

Figure 3.28 : Convergence History of Crank-Nicholson Scheme, CFL = 0.81

5. Observations

The two-step Runge-Kutta flow solver calculated the pressure distributions around the airfoils significantly faster in each test case. The advantage of the explicit scheme over the Crank-Nicholson scheme is primarily due to the Runge-Kutta scheme utilization of the vectorization capability of the workstation.

The two-step Runge-Kutta scheme convergence rate decreases significantly between one to two orders of magnitude reduction of the highest density residual. The Crank-Nicholson scheme showed a similar decrease in convergence rate for solutions of more complex flow requiring larger numbers of iterations. This decrease is related to the size of the time step used to advance the steady-state solution. The Crank-Nicholson scheme was more robust than the two-step Runge-Kutta scheme because larger CFL's were used. The low CFL limitations of the two-step Runge-Kutta scheme resulted in more iterations for the explicit flow solver than the Crank-Nicholson flow solver.

Due to its ability to solve the inviscid pressure distribution around an airfoil faster than schemes requiring large matrix inversions, the two-step Runge-Kutta scheme will be used with an optimization program on computers with vector processing capabilities for airfoil design.

IV. OPTIMIZATION USING PARALLEL PROCESSORS

Airfoil design via optimization methods require numerous CFD solutions to compute the aerodynamic performance of different airfoil geometries. In the design process, independent variables are perturbed to determine which geometry best approximates the design criteria.

For airfoil design, the designer must first select the desired performance criteria. Next, independent variables are used to describe the geometry of the airfoil's shape. After an initial CFD solution and performance evaluation are calculated, multivariable calculus determines a direction to vary the independent variables to match or optimize the performance criteria. The performance of many airfoil shapes are then evaluated using CFD, and the geometry which comes closest to the desired performance becomes the baseline solution to vary for the next optimization cycle. If there are no limitations to the allowable computer processing time, this process is repeated until an airfoil geometry is found which matches or optimizes the desired performance.

An optimization routine has been developed which divides the required performance evaluations among multiple processors. This optimization routine, when coupled with a flow solver, evaluates the aerodynamic performance of numerous airfoil geometries simultaneously and greatly decreases the time required for airfoil design.

A. OPTIMIZATION METHODS FOR MULTIVARIABLE FUNCTIONS

Elementary calculus defines a local minimum for a function of a single variable, $f = f(x)$, to be a point where the following criteria are met :

$$(1) \quad \frac{df}{dx} = 0$$

$$(2) \quad \frac{d^2f}{dx^2} \geq 0 .$$

If the first criterion alone is met, the point may also represent a local maximum or a saddle point.

For a range of x, where $x1 \leq x \leq x2$, a global minimum is the point x which corresponds to the minimum value of f(x) in that range. One or more local minima can also be found in that range as shown in Figure 4.1.

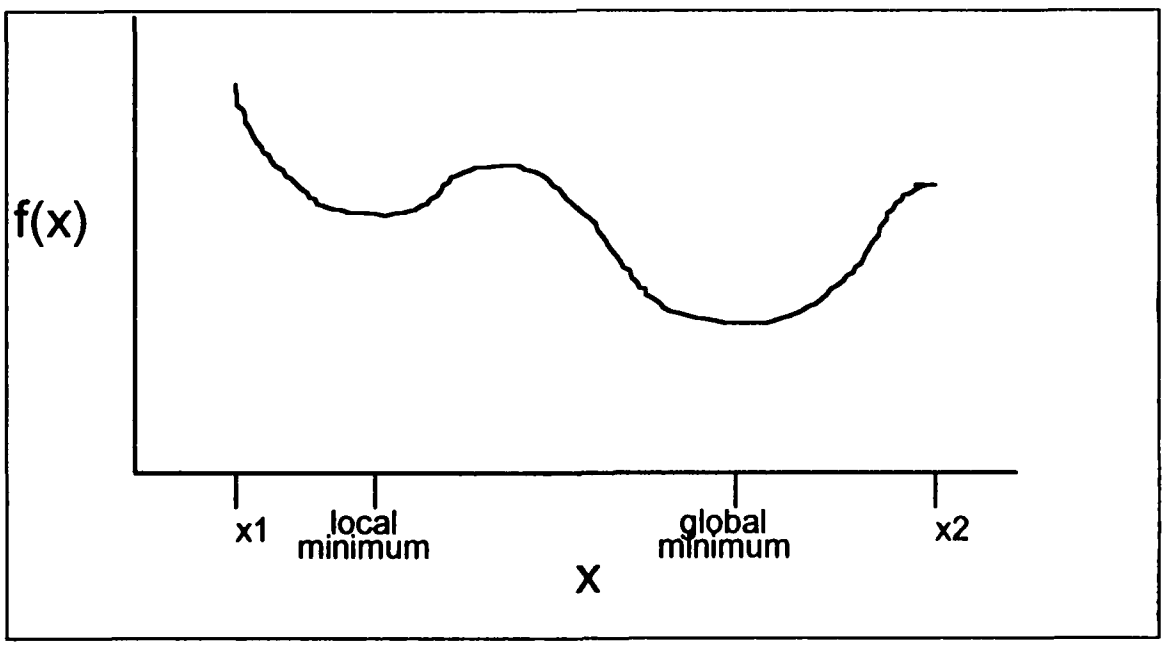

Figure 4.1 : Local and global minima for a single variable function

If f is a function of more than one variable,

$$f = f(x_1, x_2, x_3, \ldots, x_n)$$

local minima, local maxima, and saddle points are all found at points where

$$\frac{\partial f}{\partial x_1} = 0$$

$$\frac{\partial f}{\partial x_2} = 0$$

$$\frac{\partial f}{\partial x_3} = 0$$

$$\vdots$$

$$\frac{\partial f}{\partial x_n} = 0 .$$

B. OPTIMIZATION VIA NEWTON'S METHOD

An optimization technique is used in a design process to minimize an objective function which represents the difference between the actual and desired performance. The objective function, f, is a function of a set of n independent variables, **X**, placed in vector form :

$$f = f(\mathbf{X})$$

$$\mathbf{X} = [\, x_1, x_2, x_3, \ldots, x_n \,].$$

The gradient vector, **G**, is defined as a vector of the partial derivatives of the objective function with respect to **X**, or

$$\mathbf{G} = \frac{\partial f}{\partial \mathbf{X}} = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial x_3}, \ldots, \frac{\partial f}{\partial x_n} \right].$$

The Hessian matrix, **H**, is a symmetric matrix of mixed second-order derivatives of the objective function with respect to the independent variables :

$$\mathbf{H} = \frac{\partial \mathbf{G}}{\partial \mathbf{X}} = \frac{\partial^2 f}{\partial \mathbf{X}^2}$$

or

$$\mathbf{H} = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1 \partial x_n} \\ \dfrac{\partial^2 f}{\partial x_1 \partial x_2} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2 \partial x_n} \\ \dfrac{\partial^2 f}{\partial x_1 \partial x_n} & \dfrac{\partial^2 f}{\partial x_2 \partial x_n} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

For an iterative optimization method, the set of independent variables is varied slightly each iteration. A vector δ composed of small perturbations of the

independent variables is added to the set of independent variables at iteration k, where

$$\mathbf{X}^{k+1} = \mathbf{X}^{k} + \delta^{k} \tag{4.1}$$

Newton's method is used to determine the updated set of independent variables to minimize the objective function.

Newton's method is derived from a second order Taylor series for $f(\mathbf{X}^{k+1})$ centered at $f(\mathbf{X}^{k})$:

$$f(\mathbf{X}^{k+1}) = f(\mathbf{X}^{k}) + \left[\mathbf{G}^{k}\right]^{T} \delta^{k} + \frac{1}{2}\left[\delta^{k}\right]^{T} \mathbf{H}^{k}\, \delta^{k} + \text{H.O.T.} \tag{4.2}$$

The optimization method requires that the function and the components of its gradient vector and Hessian matrix be defined for each set of independent variables evaluated. Other requirements include that $f(\mathbf{X}^{k+1})$ must have a unique minimizer, and that **H** is positive definite to ensure convergence to a minimum.

The updated objective function is minimized with respect to the perturbation of the independent variables . Solving

$$\frac{\partial f(\mathbf{X}^{k+1})}{\partial(\delta^{k})} = \mathbf{0}$$

gives

$$\mathbf{G}^k + \mathbf{H}^k \delta^k = 0$$

or

$$\mathbf{H}^k \delta^k = - \mathbf{G}^k \qquad (4.3).$$

The vector δ^k gives the variation of the independent variables to minimize the objective function for the next iteration.

The basic Newton method is not suitable for a general purpose algorithm if the evaluation of $\mathbf{H}^k$ or the quadratic model of the function is not accurate. Newton's method with line search is an iterative optimization technique well suited for computational applications.

Newton's method with line search solves for a direction of search, $\mathbf{P}^k$, where

$$\mathbf{P}^k = - \left[\mathbf{H}^k\right]^{-1} \mathbf{G}^k \qquad (4.4).$$

P is a vector which gives a ratio to vary the independent variables in order to minimize the objective function. The new set of independent variables is then calculated from the relation :

$$\mathbf{X}^{k+1} = \mathbf{X}^k + q\, \mathbf{P}^k \qquad (4.5),$$

where q is a positive scalar.

After $\mathbf{P}^k$ is calculated, the objective function becomes a function of the scalar q. A directional search using Equation (4.5) to define the set of independent variables is conducted to minimize the objective function. In the directional search, q is

varied and the objective function is evaluated until a minimum is found. A new set of independent variables is then perturbed to find **P** for the next iteration.

The primary disadvantage of using Newton's optimization method with a line search is the computational time necessary to evaluate the components of the Hessian matrix **H**. Also, if the quadratic model described in Equation (4.2) is not sufficiently accurate or if **H** is not positive definite, the objective function may not decrease using a line search in the direction **P**.

C. QUASI-NEWTON OPTIMIZATION METHOD

Robert Kennelly [8] has written an optimization program, QNMDIF, which utilizes a quasi-Newton method with a line search. The main advantage of the quasi-Newton method is that the second-order partial derivative components of the Hessian matrix do not require numerous function evaluations. Instead, the Hessian matrix is estimated based upon first-order derivative calculations and is updated each optimization cycle.

QNMDIF, like Newton's method with line search, attempts to minimize a quadratic model of the objective function. The direction of search, $\mathbf{P}^k$, is determined each iteration based upon the linear system of equations

$$\mathbf{B}^k \mathbf{P}^k = - \mathbf{G}^k \qquad (4.6),$$

where **B** is an approximation to the Hessian matrix. A search is then conducted along the ray $\mathbf{P}^k$ requiring new function evaluations based upon the set of independent variables defined in Equation (4.5).

Instead of calculating the Hessian matrix which would require a large amount of processing time for the calculation of numerous expensive objective functions, the quasi-Newton method uses a sequence of matrices to form the matrix **B** to approximate **H**. After the new point $\mathbf{X}^{k+1}$ has been found and $\mathbf{G}^{k+1}$ evaluated, the Hessian approximation is modified according to the quasi-Newton relation

$$\mathbf{G}^{k+1} - \mathbf{G}^{k} = \mathbf{B}^{k+1}\left[\mathbf{X}^{k+1} - \mathbf{X}^{k}\right] \qquad (4.7).$$

Equation (4.7) would be satisfied if the objective function was quadratic and if $\mathbf{B}^{k+1}$ was the true Hessian. The approximate Hessian's symmetry and positive definiteness are preserved using a rank 2 update to **B** developed by Gill and Murray [14,pp.91-108]. The matrix **B** is represented by the factors $\mathbf{LDL}^{T}$, where **L** is a unit lower triangular matrix with a transposed matrix $\mathbf{L}^{T}$, and **D** is a positive definite diagonal matrix. In the program, **B** can first be set to the identity matrix, or the diagonal elements in **D** can be calculated using finite-difference methods requiring several function evaluations.

QNMDIF uses parabolic interpolations to estimate the minimum of the objective function in its directional search. Using this method, a multivariable objective function of n independent variables should converge to a minimum in n searches or less in the direction **P**. The quasi-Newton method does not depend upon exact line searches and is more efficient if coarse line searches are used because less function evaluations are required. This iterative scheme requires multiple function evaluations for the gradient vector calculation and a line search after each new set of independent variables are found.

QNMDIF includes several reliability features to ensure convergence of the objective function to a minimum. If a line search fails with no decrease in the objective function, the components of the gradient are calculated using central-difference approximations for the derivatives instead of forward-difference approximations. The gradient components are later calculated using forward differences if the objective function later decreases.

The optimization routine uses machine precision to evaluate if a change in the objective function is significant. If central-difference approximations for the gradient do not result in a reduction of the objective function, a sophisticated local search is applied. Line searches are conducted in orthogonal directions from perturbed positions around the set of independent variables. The local searches help protect against convergence to a local minimum or a saddle point.

Finite-difference step sizes are required for the forward or central-difference estimations of the components of the gradient vectors. The step sizes need to be large enough to produce a significant change in the objective function and small enough for accurate approximations of derivatives.

The optimization program can begin as a restart from a previous problem. Also, QNMDIF periodically updates and stores information in a restart file during an optimization problem.

D. DEVELOPMENT OF A QUASI-NEWTON OPTIMIZATION ROUTINE UTILIZING PARALLEL PROCESSING

An objective function appropriate for airfoil optimization design is based upon the aerodynamic performance of the airfoil. Objective functions necessary for the gradient calculations and line searches require numerous CFD solutions to calculate

the flow-fields around airfoils of different geometries. Therefore, the vast majority of computational time needed to design an airfoil is spent calculating the flow-fields around various airfoil geometries. A simple airfoil design test case using QNMDIF and RK2EULER requires over 96% of the processing time to be spent calculating various flow-field solutions.

A quasi-Newton optimization method was developed for aerodynamic design utilizing the Intel iPSC/860 hypercube parallel computer. The hypercube is used to simultaneously calculate the flow-fields over multiple airfoil geometries for the estimation of the gradients and in directional searches for minimum objective functions. Conducting the gradient calculations and line searches in parallel greatly increases the speed of the design procedure.

1. The Intel Hypercube Parallel Computer

The Intel iPSC/860 hypercube is a distributed memory parallel supercomputer consisting of 128 nodes with a peak performance of 7.7 gigaflops in a 64 bit architecture. The hypercube used for this research is located at the Numerical Aerodynamic Simulation laboratory at NASA Ames.

The System Resource Manager, or local host, is a UNIX machine and is used to communicate with the cube. The cube consists of 128 i860 processing nodes with 8 megabytes of memory per node. A user can utilize a group of 1, 2, 4, 8, 16, 32, 64, or 128 nodes for an application. Each node has a 40 megahertz clock and has access to the Concurrent File System.

The Concurrent File System (CFS) provides fast, simultaneous access to secondary storage for the nodes. Large data files can be written to and read from the CFS.

The iPSC/860 is a Multiple Instruction Multiple Data (MIMD) computer. An application must be designed to run in parallel in order to effectively use the hypercube. Since the iPSC/860 is a distributed memory computer, Intel provides message passing routines for communication of pertinent data between the nodes.

An application can consist of a host and a node program or just a node program. The host program runs on the local host only and communicates with the node programs. Each node of the cube executes the node program and can use different sets of data. Also, each node can perform different instructions based upon conditional statements in the program. A manager node can be used to direct the flow of information with other nodes.

The hypercube has both FORTRAN and C compilers augmented with special routines for programming in a parallel distributed memory environment. These special routines decrease internode communication time, determine individual node information, perform global operations, and write to the CFS.

Internode communications increase the computational time required in a parallel application. Communication time can be minimized by decreasing the number of messages sent. Fewer large messages can be sent faster than more numerous smaller messages. Global messages provide the most efficient communications from a single node to all other nodes assigned to an application. Global operation routines provided by Intel effectively perform mathematical operations requiring information from all nodes in an application.

Node information is also available using special routines. For an application, each node is assigned a unique identification number of 0 or higher. Intel subroutines are used to identify the number assigned each node and the total

number of nodes used in the process. These numbers can be used for conditional statements in the application.

2. A Quasi-Newton Optimization Routine for Parallel Processors

The quasi-Newton optimization program, QNMDIF, was modified for efficient application on the Intel iPSC/860 hypercube. The parallel quasi-Newton optimization subroutine, PARQNM, is written to decrease the computational processing time necessary in airfoil design by optimization. PARQNM assigns multiple processors to simultaneously calculate objective functions with different sets of independent variables. Although PARQNM was written for use with a flow solver for aerodynamic design, the optimization program can beneficially be applied to other optimization problems requiring evaluations of expensive objective functions.

PARQNM is written in FORTRAN as a node program. It makes efficient use of the Concurrent File System so that each node can separately evaluate the performance of different airfoil geometries described by different sets of independent variables. The vast majority of an airfoil design application is designed to run in perfectly parallel decomposition with each node operating on different data sets. Node information subroutines are used to generalize the number of nodes used in an application. The minimum number of nodes required for an application is dependent upon the number of independent variables of the problem.

The communication strategy is designed to minimize internode communication time. Global operations are used for simple mathematical operations when practical. Multiple single node to single node communications are done simultaneously in the routine. Also, communication time is reduced by sending large global messages instead of smaller node to node messages.

a. Overview of PARQNM

A flowchart of the optimization program is shown in Figure 4.2. The scheme for PARQNM is similar to QNMDIF, but the primary difference is the utilization of parallel processing for the evaluation of the objective functions necessary for the gradient calculation and directional line searches.

The program calling PARQNM must calculate an initial value of the objective function. The main program must send arguments for the initial value of the objective function, factors of the initial estimation of the Hessian matrix, and the original set of independent variables to the optimization program. Next, PARQNM calculates the gradient vector utilizing parallel processors. A set of linear equations is then solved to calculate the direction of search to minimize the objective function. Also, a parallel line search is performed to minimize the objective function for each new set of independent variables.

If the line search successfully reduces the objective function, the set of independent variables is updated. Then the cycle is repeated with new parallel gradient calculations and parallel line searches. When the maximum iterations are completed or the convergence criteria are satisfied, the optimization program is complete. If the directional search is unsuccessful in reducing the objective function, a local search is performed in parallel to minimize the objective function.

b. Parallel Gradient Calculation

The calculation of components of the gradient vector requires significant computational time. With the utilization of parallel processors, all function evaluations necessary for the estimation of the gradient vector can be calculated simultaneously instead of sequentially as on serial computers.

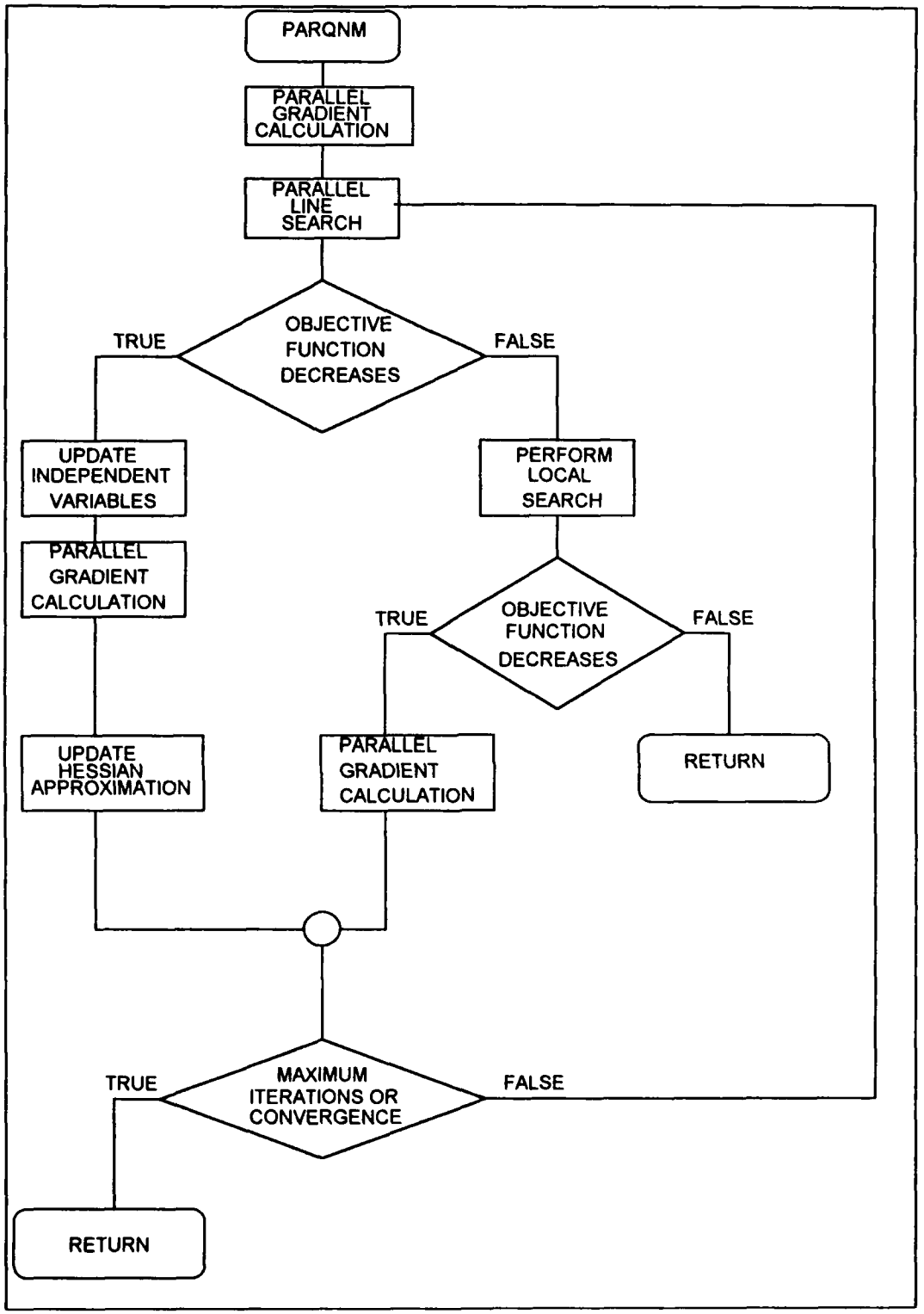

Figure 4.2 : Flowchart of PARQNM

A flowchart for the parallel gradient calculation is shown in Figure 4.3. For a second-order accurate estimation of each component of the gradient, two function evaluations are required. For example, the first gradient component is estimated from the central-difference calculation:

$$\frac{\partial f}{\partial x_1} = \frac{f(x_1 + \Delta x_1, x_2, \ldots, x_n) - f(x_1 - \Delta x_1, x_2, \ldots, x_n)}{2\Delta x_1} \qquad (4.8).$$

For n independent variables, 2n processors are used in PARQNM to calculate all function evaluations simultaneously for the estimation of the gradient vector.

Communication time is minimized by directing each processor which evaluated a function using backward-differencing of an independent variable to send its information to the corresponding processor which calculated the forward-difference estimation of the objective function. These n short messages are sent at the same time, and the components of the gradient are calculated at different processors. A manager node then collects the components of the gradient from n processors and sends the entire gradient vector to all nodes in a global message.

A comparison of the communication time required for this method versus each processor independently calculating the gradient is presented to demonstrate its advantage. For n independent variables, 2n processors on the hypercube are required for the optimization problem. A variable z is defined as $2^z = 2n$ and represents the order of the number of processors. A global message to all nodes requires approximately the same amount of communication time as z node-to-node messages of the same length.

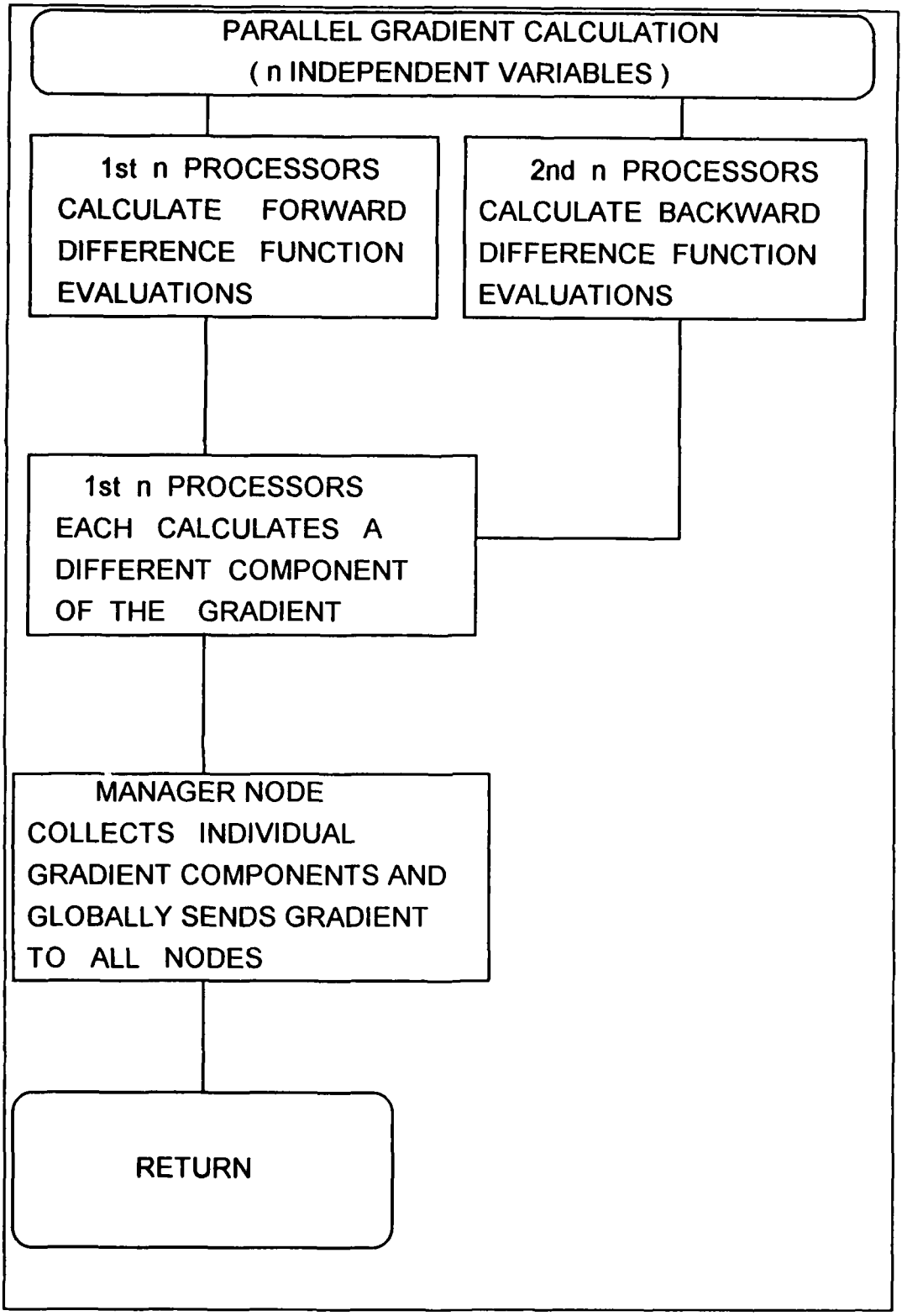

Figure 4.3 : Flowchart of Parallel Gradient Calculation

For the method described, the first n messages are sent simultaneously. Next, n-1 messages are sent to a single node followed by a global message. Therefore, the total communication time required to compute the gradient is approximately equal to the time required to pass n + z single node to single node messages. For example, a problem involving 8 independent variables requires the communication time needed to pass 12 node-to-node messages for each gradient calculation.

By comparison, if each processor performed its own calculations of the gradient, each processor would need to send its forward or backward-difference function calculation to the remaining processors in a global message. Therefore 2n global messages would be sent requiring the time to send 2nz messages. For example, an optimization problem involving 8 independent variables would require the communication time needed to send 64 node-to-node messages for each gradient calculation.

The method of gradient calculation used in PARQNM has several advantages over the method used in QNMDIF. Most importantly, all function evaluations are done simultaneously instead of sequentially. This alone typically halves the required processing time of the entire optimization routine. Also, the central-difference estimation of the parallel routine is more accurate than the forward-difference estimation of the gradient used in QNMDIF. If the forward-difference estimation of the gradient fails to provide a direction which reduces the objective function in a line search, QNMDIF will waste valuable processing time before computing the gradient vector based upon the central-difference approximations used in PARQNM.

c. Parallel Line Search

Similar to the gradient calculation, the line search used to minimize the objective function requires the evaluation of numerous objective functions corresponding to different sets of independent variables. Each objective function requires significant computational processing time. A parallel line search was developed which minimizes the objective function more efficiently and many times faster than the line search used in the sequential optimization program.

A flowchart of the parallel line search is given in Figure 4.4. After the direction $\mathbf{P}^k$ is determined, the new set of independent variables is determined from Equation (4.5). The objective function now becomes a function of the scalar q and is illustrated in Figure 4.5. For a problem, the designer selects maximum and minimum values of q. For example, in an airfoil design application, q can be assigned maximum and minimum values for the amount that the airfoil's thickness or camber can be varied in one iteration.

The line search employed in PARQNM assigns different sets of independent variables to search for a minimum objective function in the direction $\mathbf{P}^k$. Maximum and minimum values of the scalar q must be assigned for the design application. The minimum value of q, q_0, may be a small number based upon the computer's machine precision, and the maximum value of q, q_1, may depend upon the physical constraints of the problem. Different values of q are assigned in equal intervals between q_0 and q_1. Next, each processor is assigned a unique value of q and simultaneously computes the objective function for a set of independent variables based upon Equation (4.5) and illustrated in Figure 4.6. A global

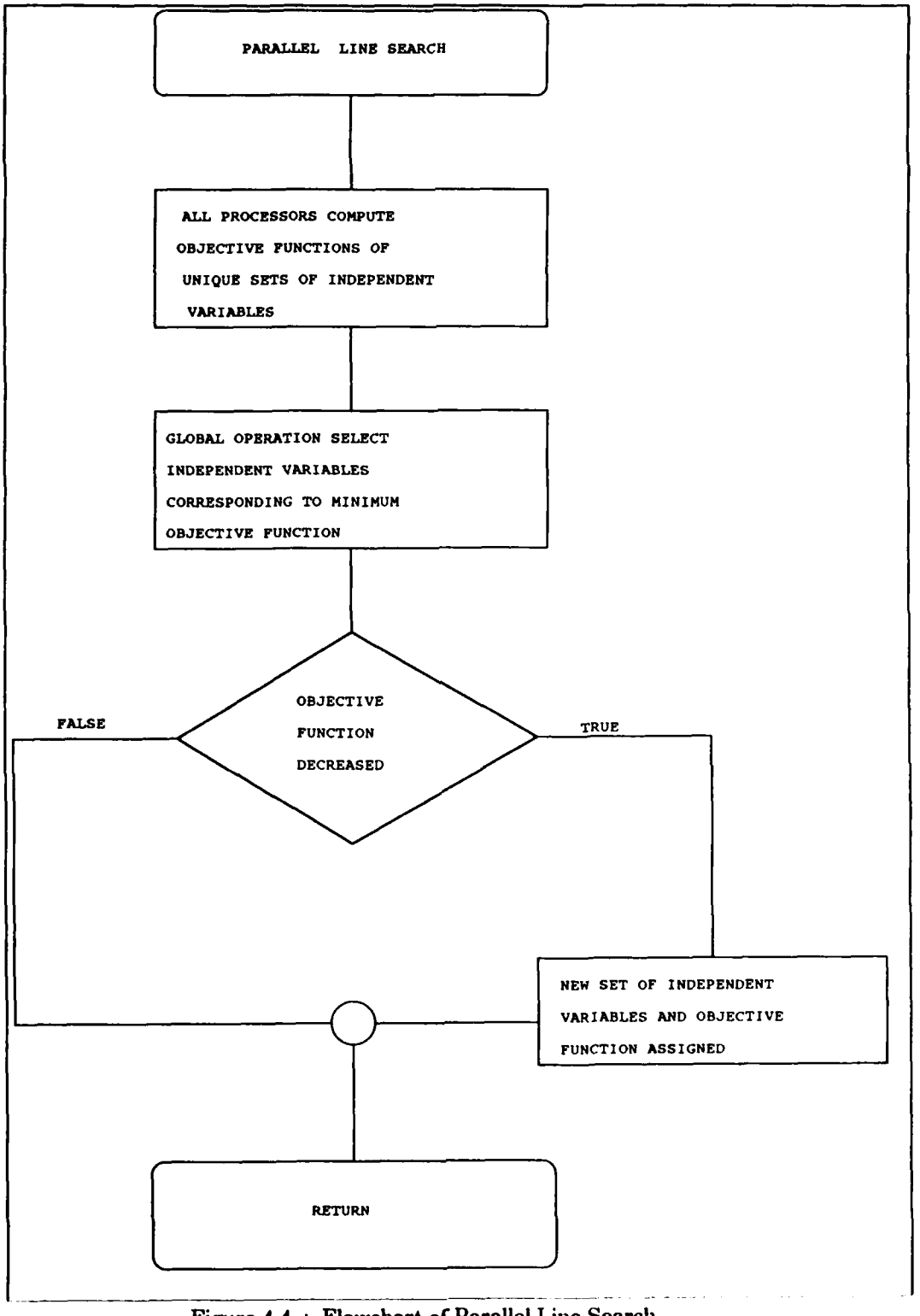

Figure 4.4 : Flowchart of Parallel Line Search

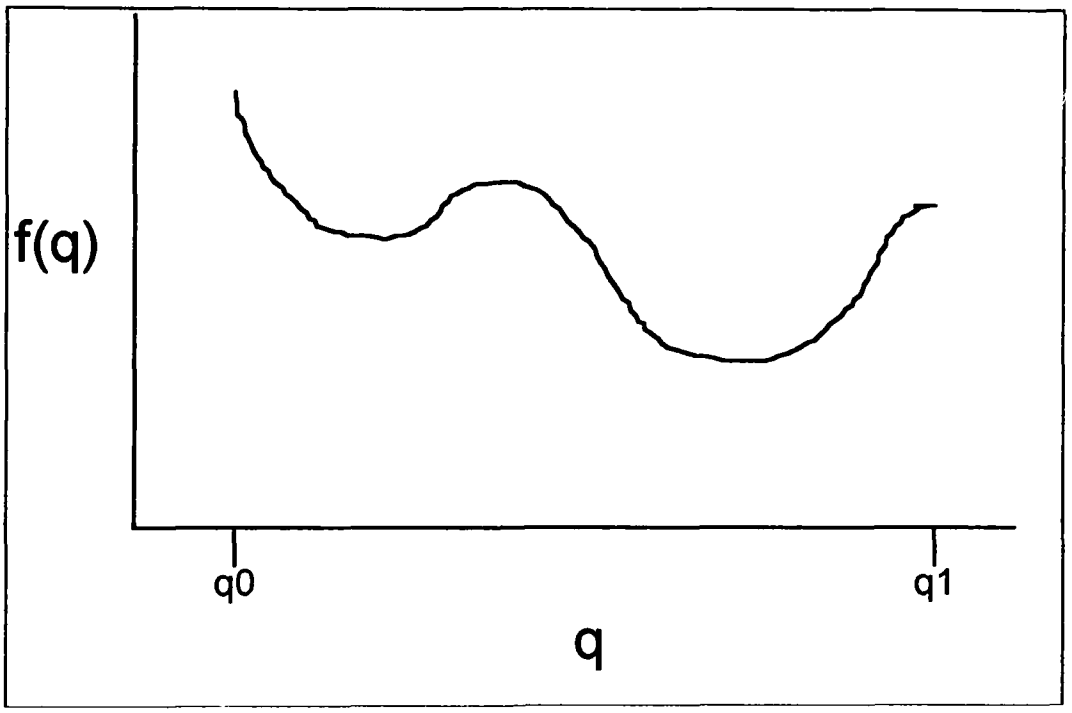

Figure 4.5 : Objective Function Dependence Upon Scalar q

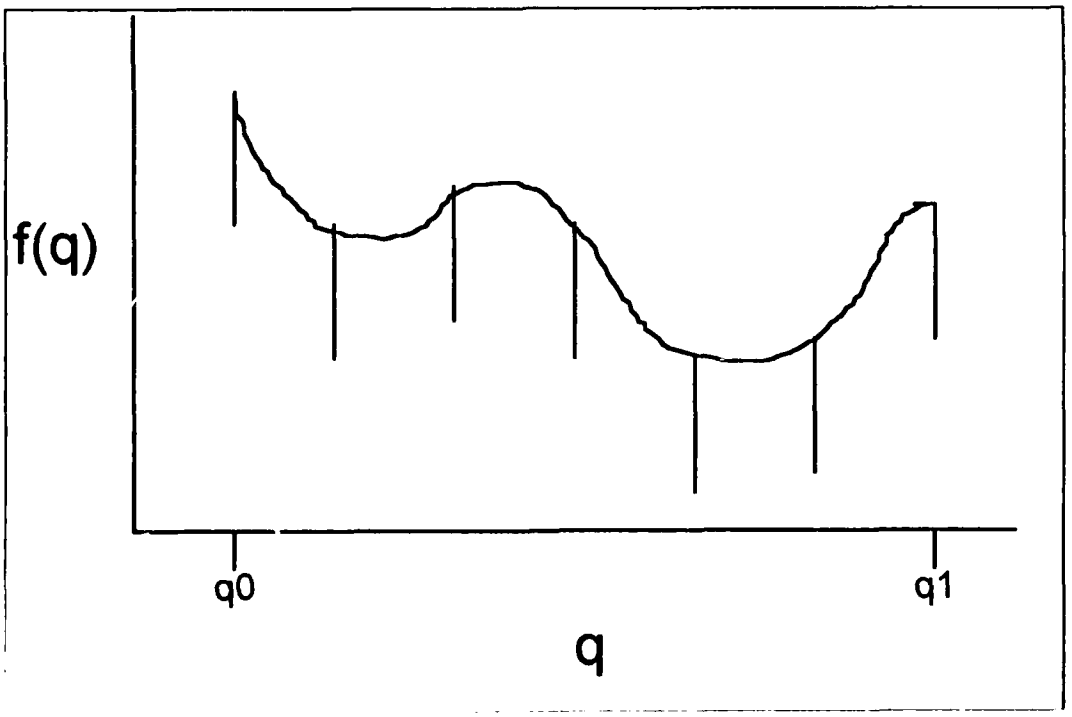

Figure 4.6 Equal Spacing Parallel Line Search

operation is used to determine which set of independent variables yields a minimum objective function. If the minimum objective function is less than the previously calculated minimum, the new set of independent variables are globally sent to all processors.

The parallel line search also has several advantages over the line search used in the sequential optimization program. PARQNM completes its line searches roughly n times faster than QNMDIF because all required function evaluations are performed in parallel. Also, the parallel line search protects against convergence to a local minimum instead of a global minimum because it searches a wide range of independent variables using equal intervals, and the accuracy of the parallel search can be improved by assigning more processors to the application. The sequential line search is more likely to converge to a local minimum because it uses parabolic interpolation for its estimation.

Some more considerations for the parallel line search are presented. Different methods of dividing the intervals between various values of q between q_0 and q_1 could have been chosen. Instead of equal intervals between different q values, an exponential distribution could have been used to evaluate more values of q closer to q_0 with larger intervals between q's assigned as the value of q increases. Also, a more thorough search could have been performed by conducting a second line search near the best value of q found in the first search. Unlike the parallel gradient routine which can utilize only 2n processors for the calculation, the parallel line search utilizes all available processors assigned to the application. If a sufficient number of processors are assigned, the fine line search divides the possible values of q among the processors. The single line search with equal spacing between q values was chosen because it provided the most effective minimization of

the objective function in the shortest amount of processing time for the airfoil optimization test cases attempted.

d. Local Search for Minimum

A local search is performed when a line search fails to reduce the objective function. The local search uses random directional searches to check whether a point can be found lower than the estimated minimum. The purpose of the search is to avoid convergence to a saddle point or a local minimum. A flowchart of the local search is given in Figure 4.7.

During the local search, two sets of line searches are conducted. A minimum is first searched in a directional search from a slightly offset vector of independent variables

$$\mathbf{Y} = \mathbf{X} + \mathbf{e} \tag{4.9}.$$

The offset vector **e** is based upon the finite step sizes used to calculate the derivatives. A directional search is first conducted in the direction $\mathbf{P}_0$, where

$$\mathbf{P}_o = - \mathbf{P} \tag{4.10}.$$

The second local search is dependent upon the success of the first. If the first local search reduces the objective function, the second search is conducted in an orthogonal direction, $\mathbf{P}_0$, from the set of independent variables **X**. If the first local search did not reduce the objective function, a directional search is conducted in the direction $\mathbf{P}_1$ where

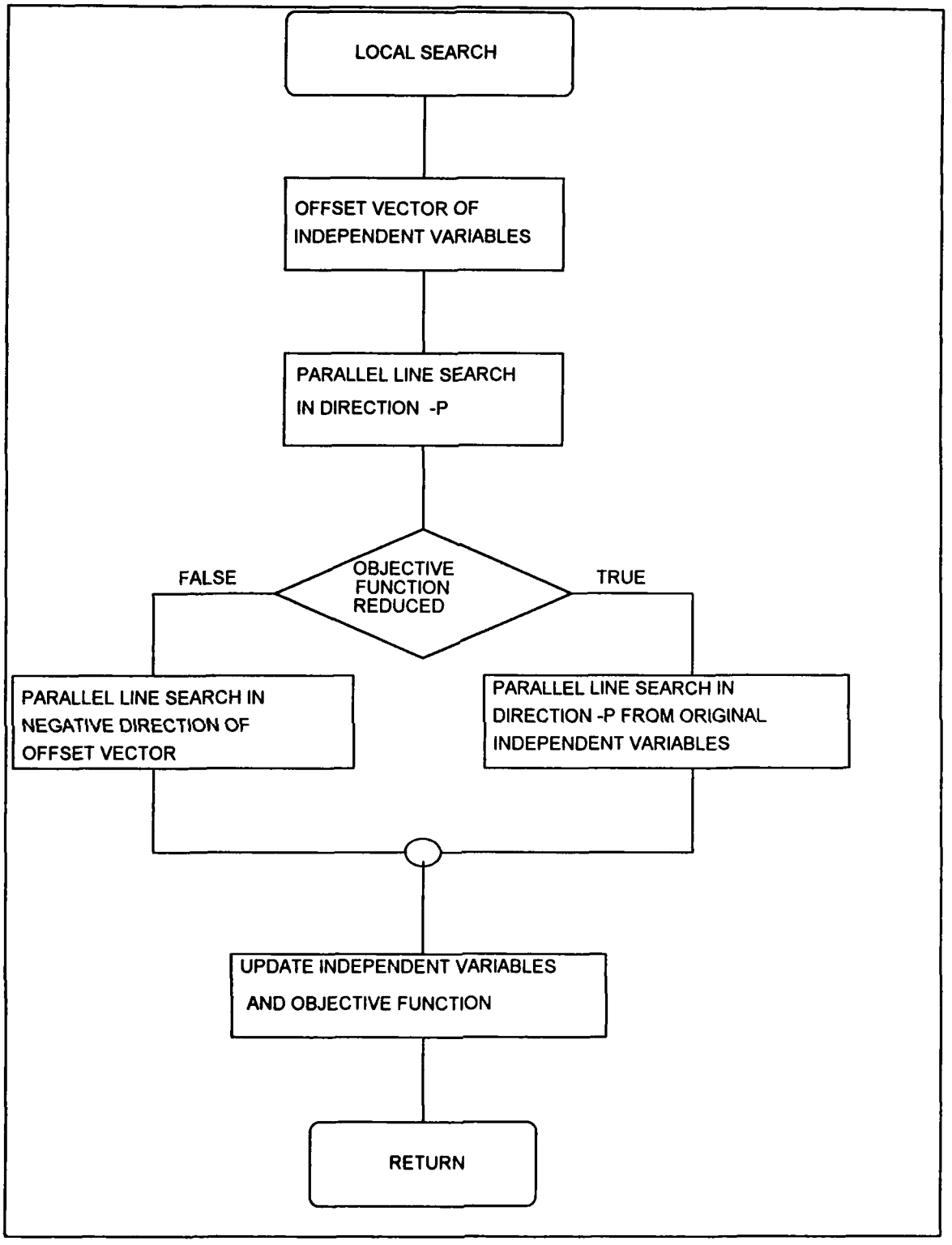

Figure 4.7 : Parallel Local Search

$$P_i = X - Y \qquad (4.11).$$

If a lower objective function is not found in the local search or if the maximum number of iterations are completed, the results are printed and information is saved in a restart file. Otherwise, the values for the objective function and independent variables are updated, a new gradient is calculated, and the quasi-Newton method is continued.

The local search of the optimization routine provides robustness against several factors including :

1. Convergence to a local minimum
2. Convergence to a saddle point
3. Inaccurate quadratic estimation of objective function in Equation (4.2)
4. Poor estimation of Hessian matrix .

Many additional objective function evaluations are required for the local search. The local search routine in PARQNM is identical to the local search used in QNMDIF except that each of the two directional searches is conducted in parallel.

A quasi-Newton optimization program with line search is a proven method to reduce an objective function dependent upon a set of independent variables. For the case of expensive objective functions, computation of the gradient and directional searches require the vast majority of computer processing time. This computational time can greatly be reduced with the utilization of multiple processors in these calculations.

E. DEVELOPMENT OF A FULLY NEWTON OPTIMIZATION ROUTINE UTILIZING PARALLEL PROCESSING

An optimization scheme utilizing a fully Newton's method with line search was developed for applications using the Intel hypercube parallel computer. The Newton method optimization scheme uses finite difference estimates of the components of the gradient vector and Hessian matrix. Then a direction of search is determined to vary the independent variables and minimize the objective function.

1. Overview

A flowchart of the Newton method optimization scheme is shown in Figure 4.8. The parallel Newton optimization scheme, PARNM, is similar to the quasi-Newton scheme except that parallel processors are used to estimate the components of the Hessian matrix in addition to the components of the gradient vector. Therefore, PARNM utilizes more processors for an application than PARQNM. Also, PARNM does not require a subroutine to update the estimation of the Hessian matrix since it is calculated directly. A parallel line search using all available processors is also used in PARNM.

2. Utilization of Parallel Processors

PARNM uses multiple processors in parallel for second-order accurate estimates of the gradient vector and the Hessian matrix. The parallel Newton's method optimization scheme calculates all objective functions necessary for estimations of the Hessian matrix and gradient vector in parallel and utilizes a Choleski decomposition scheme to calculate the direction of search with Equation (4.4). A flowchart of the calculation of the search direction is shown in Figure 4.9.

As is the case for the parallel quasi-Newton optimization scheme, the components of the gradient vector are calculated using central-differencing similar

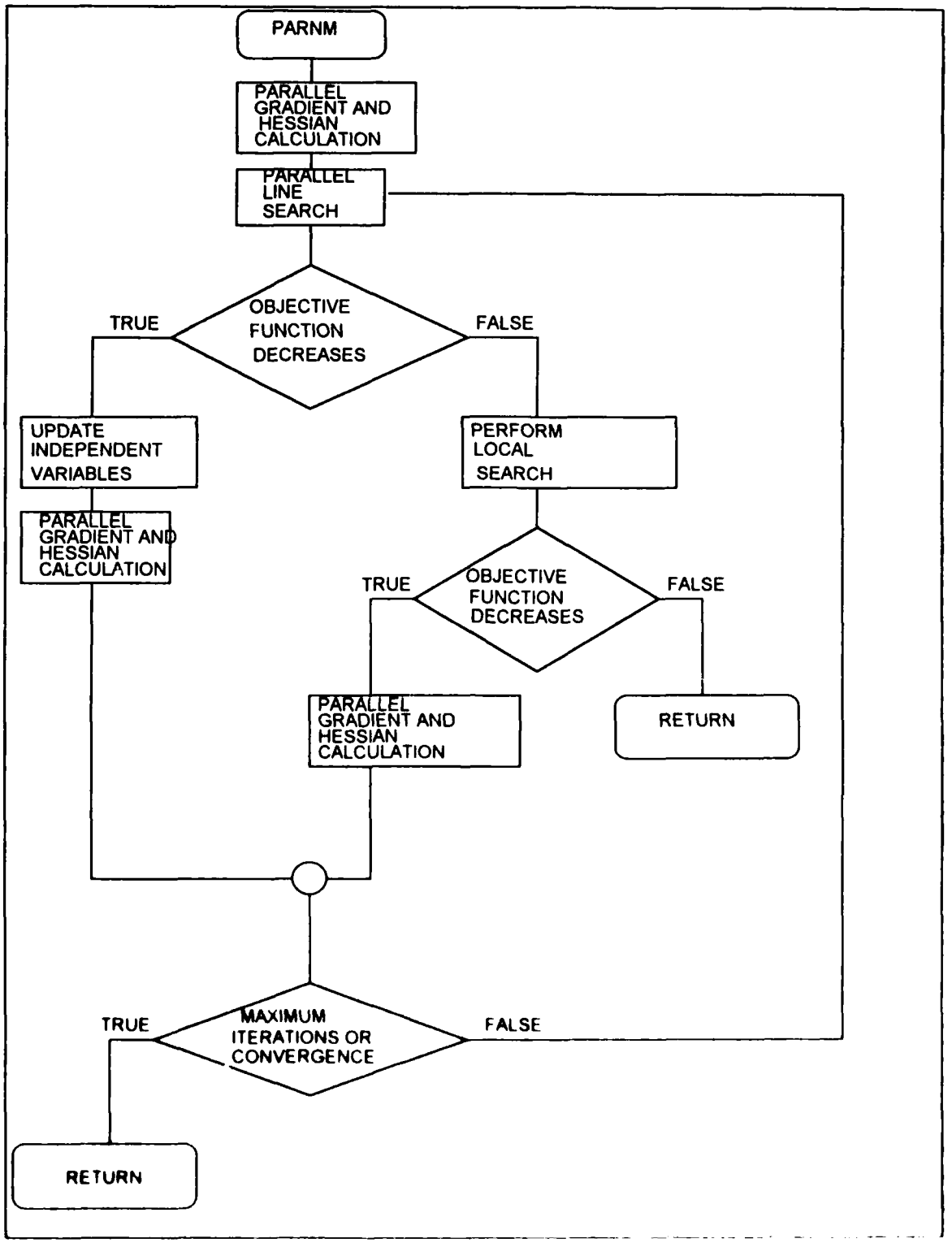

Figure 4.8 Flowchart of Fully Newton Method Parallel Optimization Scheme

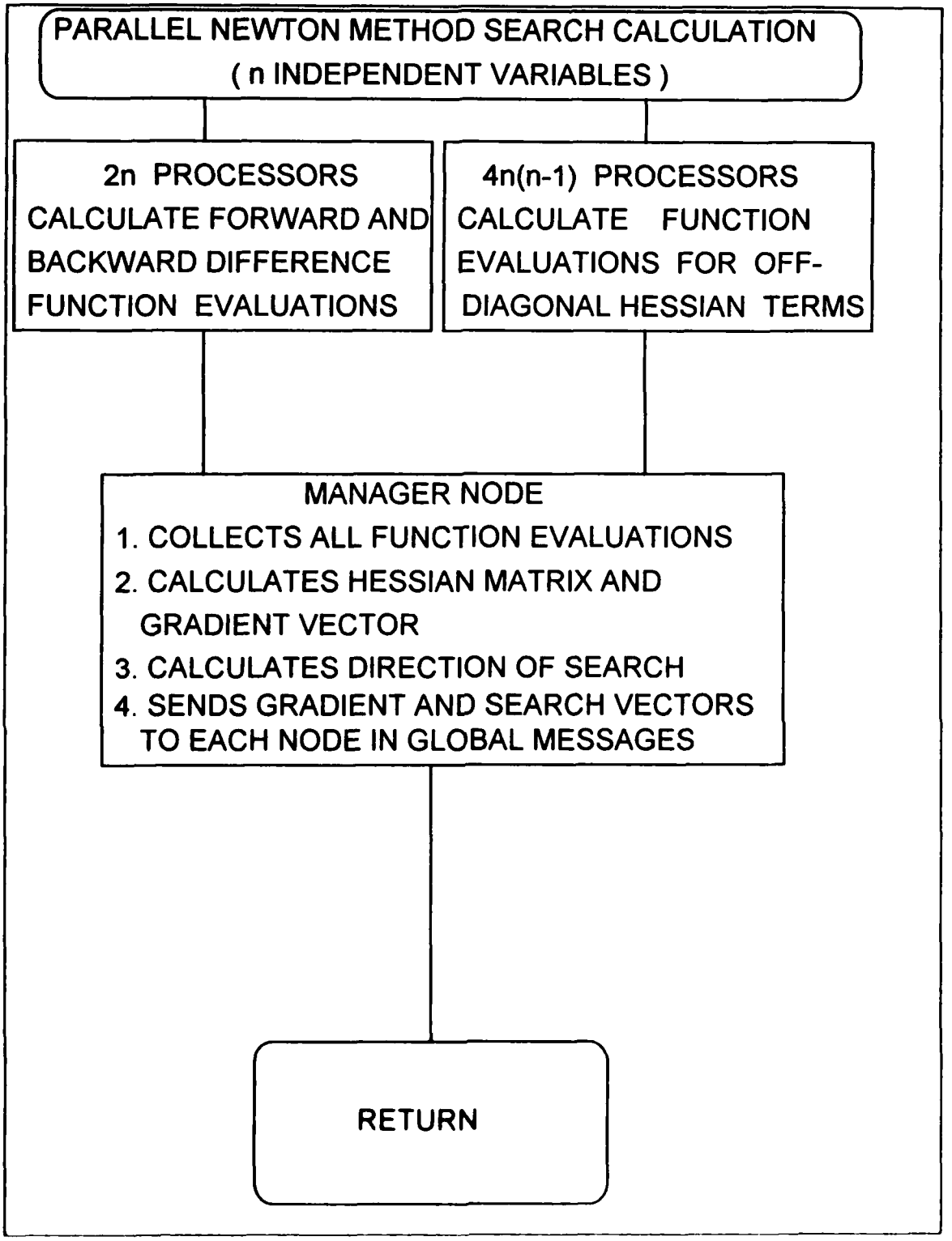

Figure 4.9 Parallel Calculation of Search Direction Using Newton's Method

to Equation (4.8). The number of processors required for an application depends upon the number of independent variables n. For a second-order accurate estimation of the gradient vector, 2n processors are required for the calculations of 2n objective functions based upon forward and backward differences using n independent variables.

The number of additional objective function evaluations required for the parallel calculation of the components of the Hessian matrix is determined by its number of off-diagonal terms. No additional objective function evaluations are required for second-order accurate estimates of the n diagonal terms in the Hessian matrix. Central-differencing is used to calculate each diagonal component of the Hessian matrix. For example, the first diagonal term is computed using the relationship

$$\frac{\partial^2 f}{\partial x_1^{\,2}} = \frac{f(x_1 + \Delta x_1,\, x_2,\, \ldots, x_n) \;-\; 2f(x_1,\, x_2,\, \ldots,\, x_n) \;+\; f(x_1 - \Delta x_1,\, x_2,\, \ldots, x_n)}{\Delta x_1^{\,2}} \quad (4.12).$$

All necessary objective function evaluations are available from the forward and backward function evaluations used in the calculation of the gradient vector plus the known objective function for the current set of independent variables.

For a symmetric n x n matrix, the total number of off-diagonal terms which needs to be evaluated is $(n^2 - n) / 2$. A typical equation for the calculation a second-order accurate estimate of an off-diagonal component of the Hessian matrix is as follows :

$$\frac{\partial^2 f}{\partial x_1 \partial x_2} = \frac{f(x_1 + \Delta x_1,\ x_2 + \Delta x_2,\ \ldots,x_n) - f(x_1 + \Delta x_1,\ x_2 - \Delta x_2,\ \ldots,x_n)}{4\Delta x_1 \Delta x_2} - \frac{f(x_1 - \Delta x_1,\ x_2 + \Delta x_2,\ \ldots,x_n) - f(x_1 - \Delta x_1,\ x_2 - \Delta x_2,\ \ldots,x_n)}{4\Delta x_1 \Delta x_2} \quad (4.13)$$

Four function evaluations are required to obtain second-order accurate estimates for each off-diagonal term because combinations of forward and backward-differences for two independent variables must be evaluated.

Therefore, $2n^2$ function evaluations are required to determine the gradient vector, Hessian matrix and subsequently the search direction for a function of n independent variables using the fully Newton optimization method. If the function evaluations are to be calculated in parallel, $2n^2$ processors must be available for the application. This limits the maximum number of independent variables to 8 which would require all 128 processors of the Intel hypercube for an application. For comparison, the parallel quasi-Newton optimization routine requires only 2n processors for an application.

3. Calculation of the Search Direction

A single manager node is used to calculate the direction of search. The parallel Newton method program allows for optimization of functions described by a maximum of eight independent variables. If more than eight independent variables are used or not enough processors are selected for an application, an error message is printed and the optimization routine stops.

The manager node receives all objective function calculations from the remaining processors used in the application. Next, the manager node calculates the components of the gradient vector and Hessian matrix using equations similar to

(4.8), (4.12), and (4.13). In order to help ensure that the search direction minimizes the objective function, the Hessian matrix is checked for positive definiteness. If a diagonal term in the estimated Hessian matrix is less than zero, then the term is replaced by a small positive value determined by the machine precision. If an off-diagonal term is calculated to be less than zero, it is replaced by a zero to ensure positive definiteness.

The gradient vector, **G**, and the upper triangular portion of the Hessian matrix, **H**, are both stored in first order arrays. An efficient Choleski decomposition routine is used to solve for the search direction **P** where :

$$\mathbf{H}\ \mathbf{P} = -\mathbf{G} \qquad (4.14).$$

The search direction and gradient vectors are then globally sent from the manager node to all other nodes.

All processors assigned to the application are then used for a parallel line search to minimize the objective function. This line search is more thorough than one from the parallel quasi-Newton optimization application due to the greater number of processors available. After a set of independent variables is found which corresponds to the minimum objective function evaluated in the parallel line search, the optimization cycle will repeat itself until the stopping criteria is met or the maximum number of iterations are completed.

F. SUMMARY

Parallel quasi-Newton and parallel Newton optimization schemes have been developed to decrease the time required to reach solutions to optimization

problems involving expensive objective functions. Parallel processors calculate multiple function evaluations simultaneously and can greatly increase the speed and efficiency of similar sequential optimization routines. The parallel Newton and quasi-Newton optimization schemes are used in airfoil design applications and compared with a sequential quasi-Newton optimization routine.

V. AIRFOIL DESIGN VIA OPTIMIZATION

A. OVERVIEW

The Newton and quasi-Newton method optimization schemes are utilized with various other programs for airfoil design. The complete airfoil optimization process requires the understanding and application of multiple disciplines including vector calculus, computational fluid dynamics, grid generation, and parallel computer programming.

The design process begins with selection of the desired aerodynamic performance for the target airfoil to optimize. An initial airfoil is chosen and its performance is evaluated. Independent variables which describe the shape of the airfoil are varied each iteration, and the performance of various shapes of airfoils are compared to determine which geometry optimizes the desired performance criteria. The hypercube parallel processing machine is the most suitable computer for airfoil design via optimization techniques because the calculation of expensive objective functions require CFD solutions which can be performed on multiple processors simultaneously.

The flowchart for an airfoil design application is given in Figure 5.1. First, files are opened on a sequential computer or the Concurrent File System of the hypercube. Next, information describing the shape of the baseline airfoil is read from a file. If the problem is a restart of a previous design problem, pertinent data is read from a restart file. A flow solver determines the aerodynamic performance of the baseline airfoil, and the objective function is then calculated. A subroutine then determines the precision of the computer running the application. Finally, the

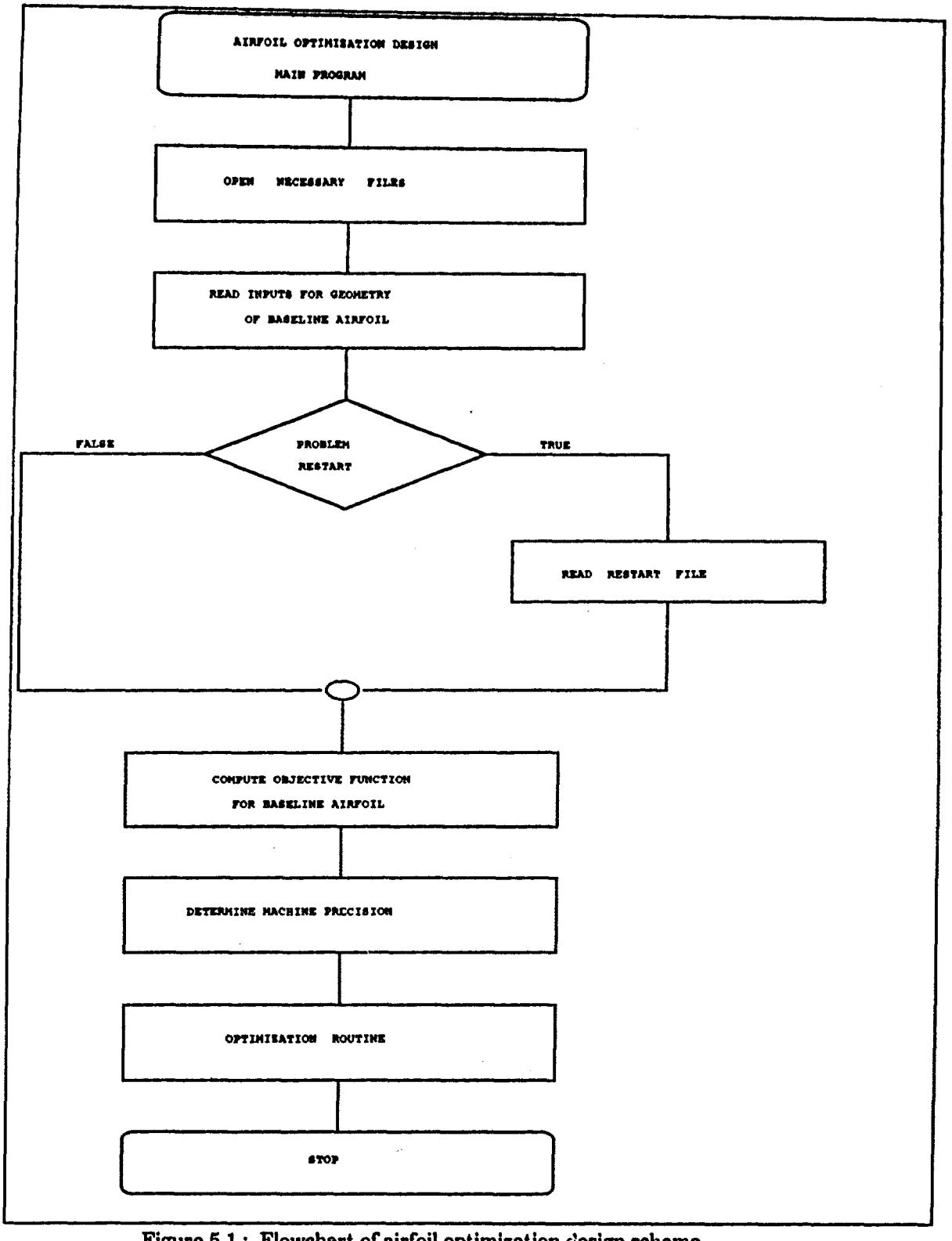

Figure 5.1 : Flowchart of airfoil optimization design scheme

optimization program uses a quasi-Newton or fully Newton method to vary and evaluate the independent variables for optimization of the desired aerodynamic performance. The airfoil design scheme will be examined with emphasis on the utilization of the Intel hypercube parallel computer.

B. UTILIZATION OF THE CONCURRENT FILE SYSTEM

The Concurrent File System is effectively utilized to manage multiple files necessary for parallel airfoil design with the Intel hypercube. The CFS assists all processors used in the airfoil design application with the parallel evaluations of the aerodynamic performance of different airfoil shapes.

Each processor assigned to the airfoil design application executes the same instructions from identical node programs. For effective use of the hypercube, the different processors need different sets of data to analyze in the design application. Utilization of the CFS is the most efficient means for the nodes to access input files and to write data into output files necessary for the application. Some of the input files are read only once during an airfoil design problem, and other input and output files are read and written into each cycle of the optimization scheme.

The CFS manages several files for the determination of the aerodynamic performance of different airfoil shapes. A single file is used by all processors to read the flight conditions for the performance evaluations of all airfoil geometries. The parallel optimization schemes assign each processor an individual grid input file containing points on the surface of a unique airfoil geometry being evaluated and points throughout the flow-field. Each processor evaluates the aerodynamic performance of its airfoil shape with the flow solver assigned to the application.

The output from the flow solver executed by each processor includes a file containing the flow-field properties and a file containing aerodynamic performance information such as the pressure distribution around the airfoil. These output files from each processor are assigned file numbers including its processor identification number and are written into the CFS. During each cycle of the optimization scheme, new grid files are assigned to each processor. The information from the output files is used to evaluate the objective function for the assigned airfoil shape.

C. DESIGNATION OF AIRFOIL GEOMETRY

The geometry of an airfoil is defined by a set of independent variables placed in vector form. These independent variables are varied during the optimization process, and the aerodynamic performance of each subsequent airfoil shape is evaluated. A geometry package developed by Verhoff, Stookesberry and Cain [15] of the McDonnell Douglas Corporation for airfoil optimization design is used to designate the independent variables for the application.

An airfoil geometry can be defined by a thickness distribution and a camber distribution along its chord. The position of each point on the airfoil surface is found by the addition and subtraction of the thickness from the mean camber line along the chord. The geometry package evaluates the coefficients of Chebychev polynomials for the representation of the thickness and camber distributions of the airfoil.

Chebychev polynomials possess excellent qualities for use in the representation of the thickness and camber distributions. Each point of the thickness and camber distributions is defined by a Chebychev polynomial in the form

$$y = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots + a_v x^v \quad ,$$

where x is the chordwise coordinate, y is the coordinate perpendicular to x, and v is the order of the polynomial. The accuracy of the representation generally improves with an increase in the order of the polynomial. All coefficients of a Chebychev polynomial are orthogonal to each other. Therefore, the addition of higher order coefficients to describe a distribution does not effect the lower order terms. For most cases, only a few coefficients are necessary for an accurate representation of the shape of an airfoil.

Several options are available with the geometry package for the choice of independent variables to be used in the application. The independent variables can be designated the coefficients of the Chebychev polynomials representing the thickness and camber distributions of the airfoil. Also, the independent variables can be the collocation points along the thickness and camber distribution, and the geometry package then calculates the coefficients of the Chebychev polynomials through interpolation.

Additionally, the airfoil can be described by the upper and lower surfaces instead of the thickness and camber distributions. For this case, the independent variables are designated as the Chebychev coefficients or the collocation points for both surfaces.

The geometry package must be used to describe the baseline airfoil from which to begin the optimization process. The geometry package contains algorithms to describe NACA 4, 5, and 6 series airfoils. Also, the Chebychev coefficients describing an airfoil can be input to define the baseline airfoil.

The primary advantage of using Chebychev polynomials for representation of the airfoil is the relatively few number of independent variables required for the optimization process. With fewer independent variables, fewer objective function

evaluations are required by an optimization routine each iteration. The orthogonality of the Chebychev coefficients helps ensure that the selected independent variables are truly independent of each other, which is an important assumption in the formulation of the optimization methods. Also, utilization of the Chebychev polynomials provide for smooth airfoil profiles.

D. OPTIMIZATION RESTART FILE

If the optimization application is a restart of a previous design problem, optimization data is read from a restart file. Data which is read includes the values of the independent variables, the gradient vector, components of the estimated Hessian matrix, and the last evaluated objective function. The objective function is then recomputed for the independent variables read, and a warning is printed if it does not correspond with the value read from the restart file. Information is updated into this file following the completion of the optimization process.

E. EVALUATION OF THE OBJECTIVE FUNCTION

The objective function is first computed for the baseline airfoil and then recomputed multiple times for the calculation of the gradient components and in the directional search for a minimum. The objective function represents the difference between the actual airfoil performance and its desired aerodynamic performance. A flow solver is required to evaluate the properties of the flow-field around the airfoil which is used to determine its aerodynamic performance. In airfoil design, the vast majority of processing time is used solving the flow-field around various airfoil shapes for the computation of their objective functions.

1. Overview

A flowchart of the evaluation of the objective function is given in Figure 5.2. The number of independent variables and variations to the independent variables are sent to a subroutine. The variations are added to the independent variables, and the geometry package is used to define the surface of the airfoil. Next, a grid is generated around the airfoil. A flow solver then evaluates the flow-field around the airfoil, and the objective function is calculated based upon the desired performance criteria.

2. Variation of the Independent Variables

Objective functions are calculated for the baseline airfoil, for the estimation of the gradient vectors, and in directional searches. For the calculation of the objective function of the baseline airfoil selected for the optimization process, the independent variables are normally not varied. However, the baseline airfoil can be varied with the variations written into an input file.

The independent variables are varied in the estimation of the gradient vector. In the parallel gradient calculations, each processor varies a single independent variable either a forward or backward finite difference step size to estimate different components of the gradient vector.

In the parallel directional search, the direction of the variation, **P**, is calculated using the quasi-Newton or fully Newton optimization method. The variation of the independent variables for each processor is determined by multiplying **P** by a unique scalar value q. Then the variations are added to the set of independent variables as shown in Equation (4.5).

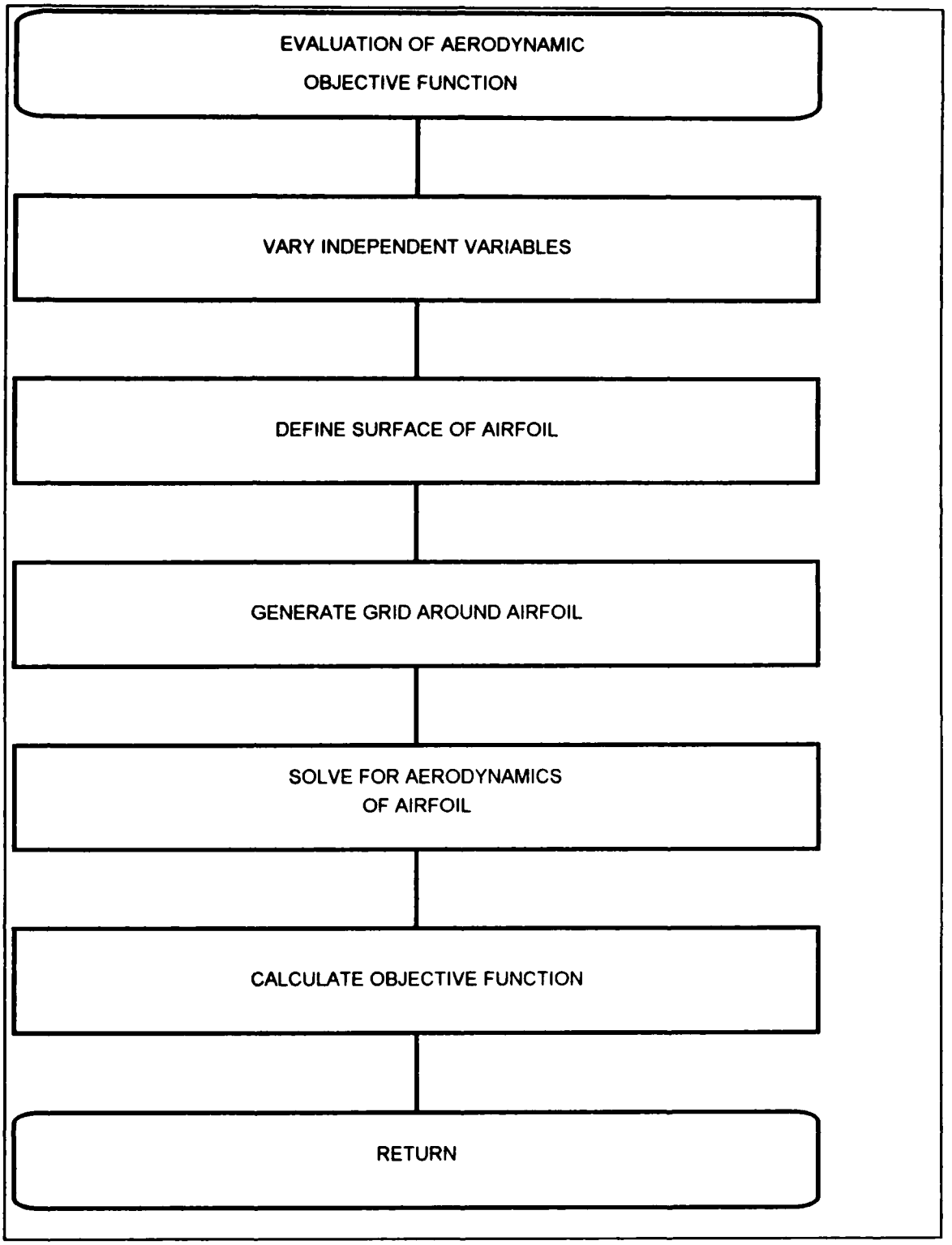

Figure 5.2 : Calculation of Objective Function for Airfoil Design

3. Grid Generation

Grids are generated around the airfoil surface using the program GRAPE [13]. GRAPE was modified for use as a subroutine so that each processor can generate numerous grids throughout the design application. The geometry package developed by the McDonnell Douglas Corporation [15] is used to define the surface of the airfoil based upon the independent variables. The coordinates of the surface of the airfoil are then read into an input file for GRAPE.

4. Calculation of the Objective Function

The calculation of the objective function needs to be based upon the aerodynamic performance of the airfoil. The optimization program is written to minimize the objective function. If the desired criteria needs to be maximized, the objective function must represent a value to be minimized.

A common airfoil design criterion is to match or optimize a desired pressure distribution around an airfoil. The target pressure distribution is read from an input file. The objective function, f, is calculated by summing the square of the difference between the desired and actual pressure distribution around the airfoil, or in equation form :

$$f = \sum_{itel}^{iteu} \left(C_p^{\,target} - C_p^{\,actual} \right)^2$$

where itel and iteu are points at the lower and upper trailing edges respectively .

Many other possible objective functions can be incorporated into the design program. In order to maximize lift, the objective function is defined as the reciprocal of the square of the lift coefficient. Other objective functions can be

defined by combinations of performance criteria based upon the airfoil's pressure distribution, lift, drag, and moment coefficients. Physical constraints upon the geometry of the airfoil can also be incorporated into the objective function including

- maximum thickness
- location of maximum thickness
- trailing edge angle
- maximum camber
- location of maximum camber
- minimum leading edge radius
- minimum volume .

The design criteria and objective function selected depend upon the flow-field properties computed by the flow solver. The flow solver RK2EULER can be used for design applications such as the optimization of an inviscid target pressure distribution at set flight conditions. If viscous effects are to be included in the design criteria, another flow solver must be used.

The optimization process involves only the objective function and is not directly dependent upon the flow solver. Therefore, the optimization schemes can be coupled with different flow solvers for different applications. The selection of the most suitable flow solver to be used for an airfoil design application depends largely upon the design criteria for the problem. The versatility of optimization design methods to use different flow solvers in airfoil design problems is not available with inverse airfoil design techniques.

F. DETERMINATION OF MACHINE PRECISION

The determination of machine precision is important for use in the calculation of the finite difference step sizes. In addition, the calculated machine precision is used as a measure of the smallest significant change in the objective function. A routine written by Forsythe, Malcolm, and Moler [16] is utilized to determine a value ε to represent the precision of the processing machine. This value, ε, is an approximation for the smallest quantity such that

$$(1 + \varepsilon) > 1$$

in floating point arithmetic.

G. SUMMARY

The parallel quasi-Newton and parallel fully Newton optimization schemes have been incorporated into an airfoil design program. Advantages for the use of an optimization method in airfoil design include the various performance criteria and flow solvers which can be used in a design application. The utilization of multiple processors of the hypercube parallel computer for simultaneous CFD solutions of different airfoil shapes significantly reduces the required processing time. The presented design procedures was evaluated for test cases with known solutions and used for actual design cases. These results are presented in the next chapter.

VI. RESULTS

A. OVERVIEW

Two test cases and two airfoil design applications were performed utilizing optimization schemes coupled to a flow solver. The test cases were constructed to design an airfoil to match the pressure distributions corresponding to airfoils of known shapes. If successful, the airfoils designed in the test cases should approach the shapes of the airfoils used to calculate the target pressure distributions. The results of the first test case were used to compare the solutions found using the sequential and parallel optimization programs.

Airfoil design applications were completed utilizing a parallel optimization routine. One design application involved transonic flow and another used an internal flow Navier-Stokes flow solver to design a symmetric cascade blade to minimize viscous losses while maintaining adequate volume in the blade for cooling purposes. These test cases and design applications demonstrate the practicality, versatility, and possible design utilizations for aerodynamic design via optimization using parallel processors. Also, these results demonstrate the necessity of the designer's intervention in an optimization application to ensure a practical solution.

B. TEST CASE 1 : SUBSONIC, NON-LIFTING AIRFOIL DESIGN

The first test case varied the thickness of a symmetric airfoil to match the pressure distribution around a thicker symmetric airfoil in subsonic flight conditions. Also, the airfoil was set at no angle of attack for further simplification of the problem. This test case utilized both parallel optimization schemes and the

sequential version. First, the parallel and sequential quasi-Newton optimization schemes are compared, and then the parallel Newton scheme is evaluated.

The goal of this test case was to use the optimization routines to design an airfoil to match the inviscid pressure distribution around a symmetric airfoil. The design pressure distribution was calculated using RK2EULER to solve the flow-field around a NACA 0012 at no angle of attack with a freestream Mach number of 0.6.

1. Baseline Airfoil and Independent Variables

The baseline airfoil used in this application is a NACA 0008 symmetric airfoil. Thicknesses of the airfoil at eight different positions along the chord of the airfoil were chosen as the independent variables. Five thicknesses were varied at positions in the forward third of the airfoil where the largest pressure changes occur. The geometry package was utilized to fit a ninth-order Chebychev polynomial through the collocation points to describe the airfoil's thickness distribution. The coefficients of the polynomial describing the camber line were set to zero for this test case. The airfoil shapes were perturbed slightly each optimization cycle for the calculation of the gradient vector and for the directional search to minimize the objective function.

2. Grid Generation

The geometry package computed the points on the surface of the airfoil. Next, the GRAPE subroutine generated a grid of 133 x 34 points around the airfoil. The grid file was then read by the flow solver.

3. Flow Solver

The flow-field properties and aerodynamic performance of various airfoil shapes were calculated by the two-step Runge-Kutta Euler flow solver RK2EULER. Experience with airfoil design showed the importance of accurate aerodynamic

performance evaluations by the flow solver for use in the calculation of the objective function. The number of flow-field iterations was set to ensure complete CFD solutions and to ensure that parallel flow-field evaluations were performed in approximately the same amount of time. The performance of the NACA 0008 baseline airfoil was calculated using 2000 iterations of RK2EULER with the flow-field initialized at the freestream Mach number of 0.6. The Mach contours for this calculation are shown in Figure 6.1. The following airfoil shapes were calculated using only small thickness variations of no more than 1% of the airfoil's chord, and the flow-field properties were initialized with values of the previous flow-field solution. These subsequent flow-field evaluations required a fewer number of flow-field iterations to reach the steady-state CFD solutions which decreased the amount of processing time. After a complete baseline solution is obtained, all subsequent performance evaluations were calculated in 600 flow-field iterations with the flow-field initialized with the values of the previous solution.

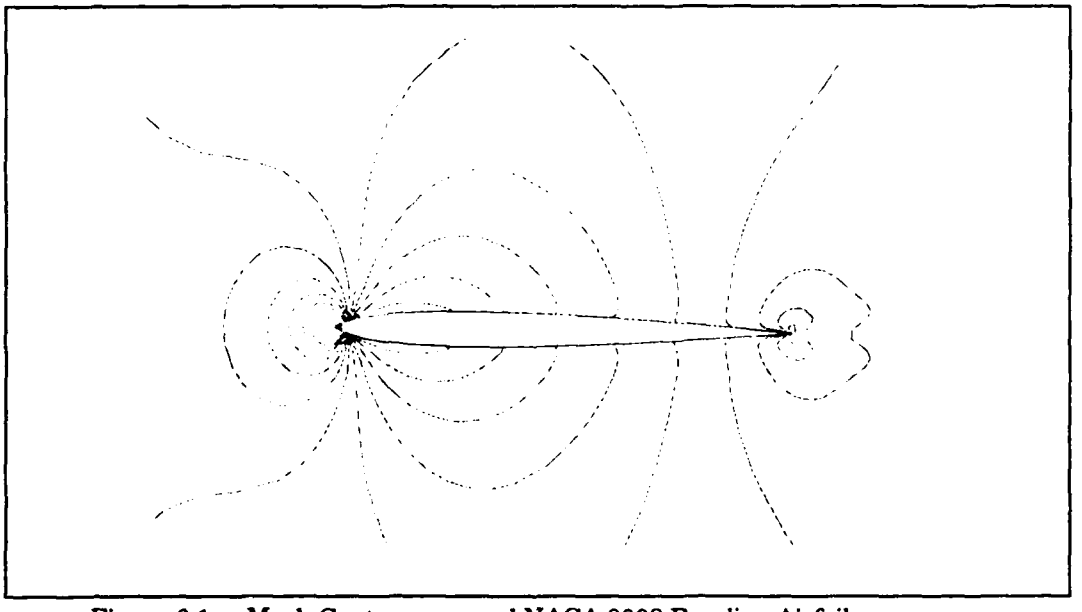

Figure 6.1 : Mach Contours around NACA 0008 Baseline Airfoil

4. Performance Criteria

The Mach contours around the target airfoil are shown in Figure 6.2. Also, the pressure distributions for the baseline and target airfoils are shown in Figure 6.3; only two curves are shown because of the symmetry of the flow around the upper and lower surfaces over both airfoils. The desired coefficient of pressure at 73 points around an airfoil was read from an input file.

The objective function associated with each airfoil shape was determined by summing the square of the difference between the desired and calculated coefficients of pressure,

$$f = \sum_{i=1}^{73} \left(C_{p\text{-calculated}_i} - C_{p\text{-target}_i} \right)^2 ,$$

where the 73 points were located around the airfoil surface.

5. Stopping Criteria

The parallel quasi-Newton optimization routine, PARQNM, and the sequential quasi-Newton version, QNMDIF, were first used in the airfoil design process for comparison. Both routines were instructed to quit based upon the same criteria. A successful optimization application would be completed when the objective function was reduced to a value less then 10% of the objective function calculated for the baseline airfoil. Otherwise, the application would quit after 20 optimization cycles were completed.

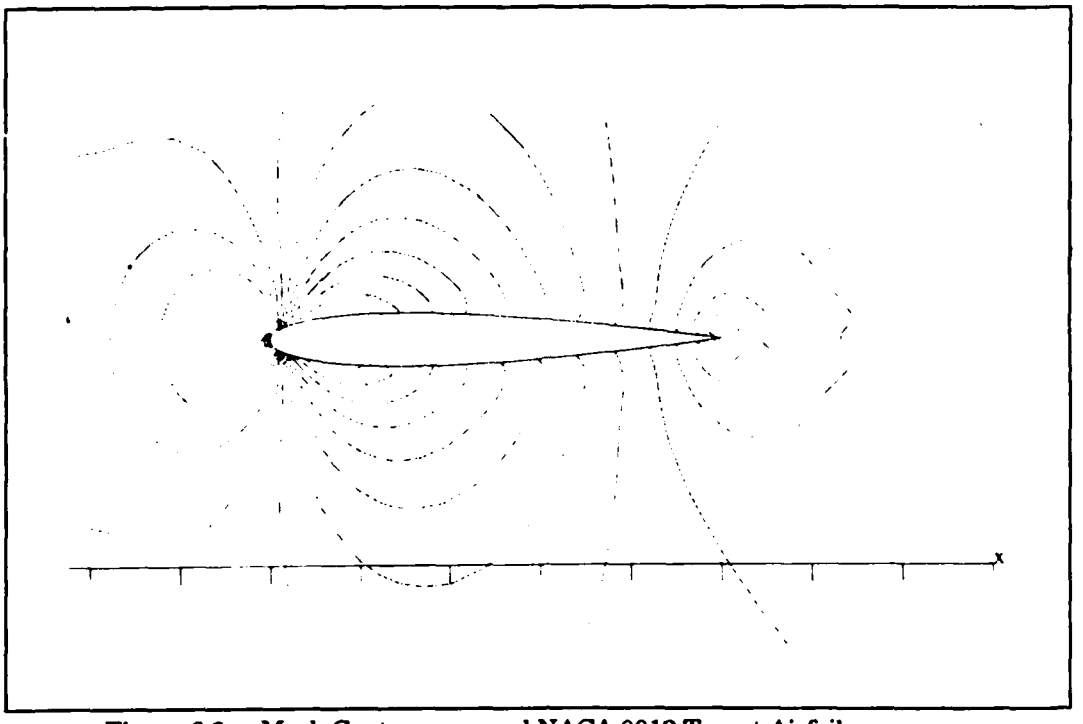

Figure 6.2 : Mach Contours around NACA 0012 Target Airfoil

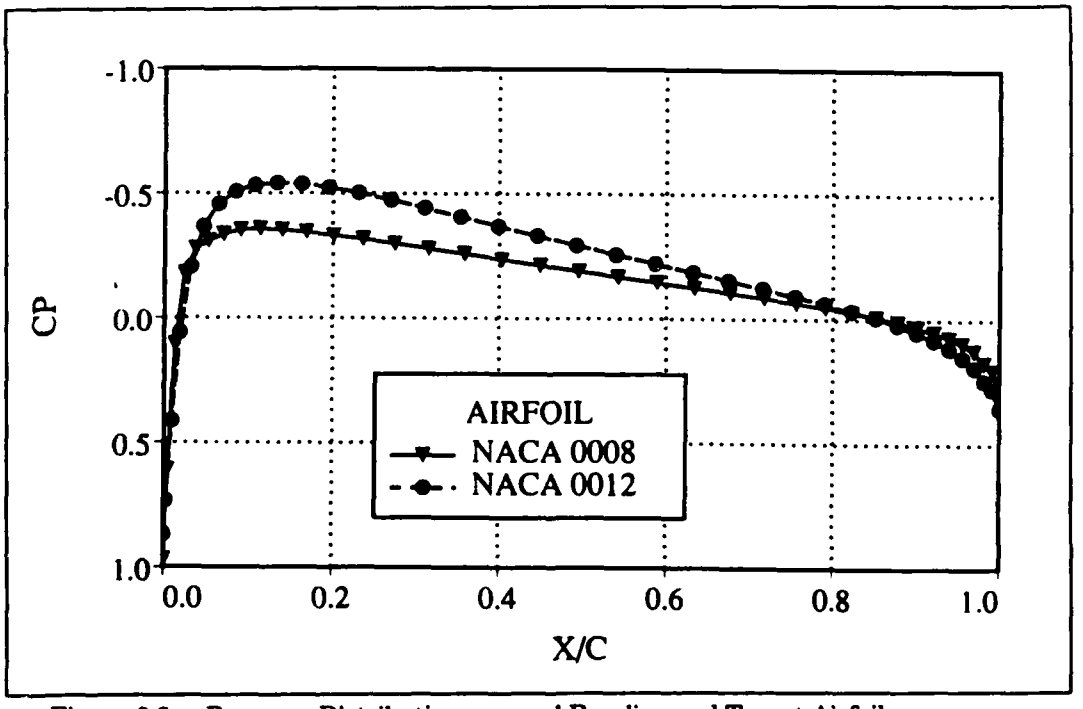

Figure 6.3 : Pressure Distributions around Baseline and Target Airfoils

6. Computers

The parallel quasi-Newton optimization design was performed using sixteen processors on the Intel iPSC/860 hypercube computer. Sixteen processors were necessary because the calculation of each of the eight gradient components required two processors for the parallel estimation of two objective functions. The sequential optimization design was performed using a UNIX workstation with a single i860 processor.

7. Comparison of Sequential and Parallel Quasi-Newton Schemes

The baseline airfoil objective function was calculated to be a value of 0.831. The parallel quasi-Newton optimization routine reduced the objection function more efficiently each optimization cycle than the sequential routine, and only the parallel optimization routine decreased the objective function to less than 10% of its original value. Also, PARQNM reached the stopping criteria in a small fraction of the processing time required for airfoil design using QNMDIF to reach its maximum number of iterations.

Table 6.1 compares the performances of the two optimization routines. The parallel optimization scheme completed the airfoil design test case approximately 18 times faster than the sequential case and in a fewer number of optimization cycles. The utilization of parallel processors significantly decreased the processing time necessary for the airfoil design test case and increased the efficiency of the optimization scheme.

Table 6.1 : Comparison of Quasi-Newton Optimization Routines

Baseline Airfoil : NACA 0008

Target Airfoil : NACA 0012

M = 0.6, AOA = 0 degrees

OPTIMIZATION ROUTINE	OPTIMIZATION CYCLES	FINAL OBJECTIVE	CPU TIME (HR:MIN:SEC)
QNMDIF	20	0.124	72:06:53
PARQNM	7	0.064	4:01:20

Figure 6.4 compares the convergence history of the objective function using the two quasi-Newton optimization schemes. After 4 optimization cycles which consisted of 5 parallel CFD solutions, the parallel quasi-Newton optimization scheme reduced the objective function at a steady rate and decreased the objective function to 26% of its initial value. The following 3 optimization cycles reduced the objective function at a slower rate to a final value of 7.7% of its initial value. This run required each of the 16 processors to calculate 8 objective functions in parallel and was completed in roughly 4 hours.

In comparison, the sequential optimization scheme showed a much lower convergence rate and did not reach the successful stopping criterion after completing 20 optimization cycles. After 4 optimization cycles including 73 flow-field evaluations by a single i860 processor, the objective function was only reduced to 84% of its initial value. The final objective function was reduced to 15% of its initial value after completing 407 flow-field evaluations.

The parallel airfoil design was more efficient than the sequential airfoil design because of the utilization of multiple processors for the parallel gradient calculation and the parallel directional searches. The parallel optimization scheme used central-difference estimations of the derivatives for the calculation of each

component of the gradient vector. The sequential optimization scheme would first attempt to use forward-difference estimations of the derivatives because less objective function evaluations would be required. If the directional search used by QNMDIF did not reduce the objective function based upon the forward-difference estimation of the gradient, QNMDIF would then have to recompute central-difference estimations and conduct another directional search. This resulted in great inefficiencies for the sequential airfoil design test case, especially during the early stages of the design process when central-difference estimations were necessary to decrease the objective function.

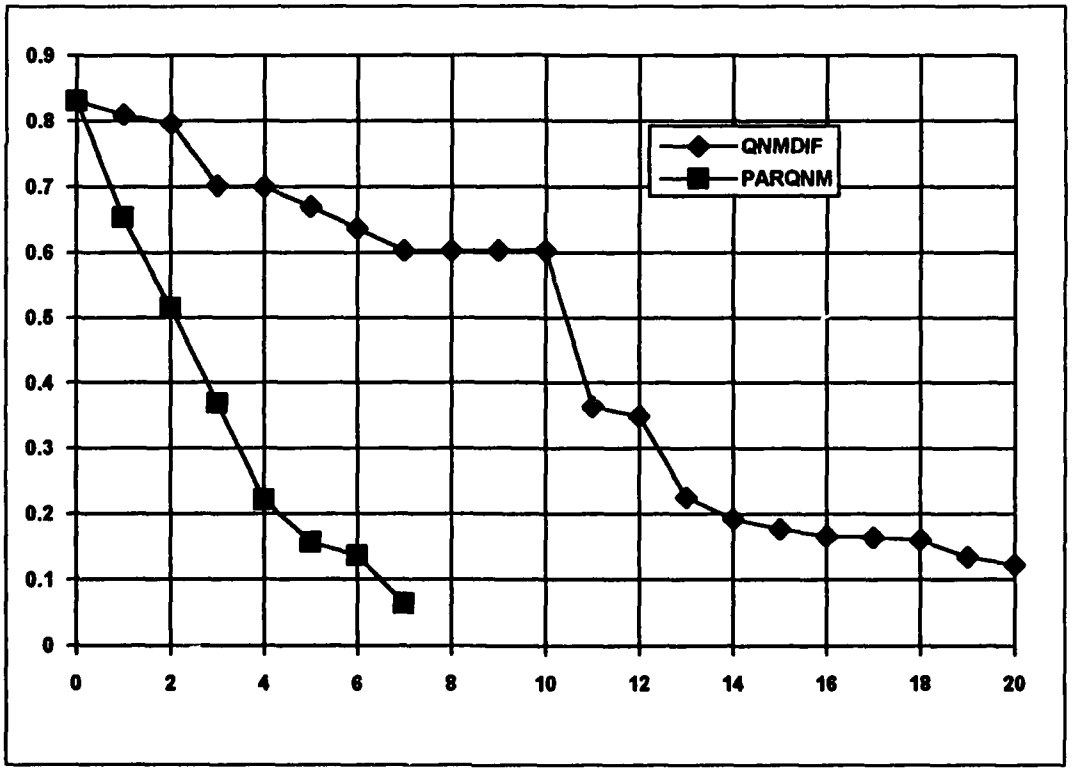

Figure 6.4 : Convergence of Sequential and Parallel Optimization Schemes

Also, the parallel line searches were more efficient in reducing the objective function than the sequential line searches. The minimum variation for directional searches were determined by the value of the estimated machine precision, and the maximum variation of the thickness was set to 1% chord to ensure only small perturbations of the airfoil shape. These values were used by both optimization schemes. Each sequential line search evaluated a maximum of eight objective functions in the direction of search and estimated a minimum using parabolic interpolation. The parallel line searches were more thorough than the sequential searches because they evaluated 16 objective functions in the direction of search including the maximum and minimum variations. Also, a local search was required to decrease the objective function with the sequential scheme after an unsuccessful directional search. Local searches were not required for the parallel optimization scheme which decreased the objective function after each directional search.

The baseline airfoil geometry and the geometries of the design airfoil after three and seven optimization cycles are shown in Figure 6.5. The baseline airfoil is the thinnest airfoil shown, and the thicknesses along the design airfoil increased each optimization cycle. Their corresponding pressure distributions are shown in Figure 6.6. As the optimization scheme varied the shape of the airfoil, its pressure distribution approached the target pressure distribution.

The final shapes and resulting pressure distributions of the design and target NACA 0012 airfoils are shown in Figure 6.7 and Figure 6.8 respectively. The design airfoil's shape and pressure distribution are nearly identical to those of the target airfoil from the leading edge to the point of maximum thickness where the majority of the thicknesses were varied. The remaining shape of the design airfoil remains thinner than that of the target airfoil.

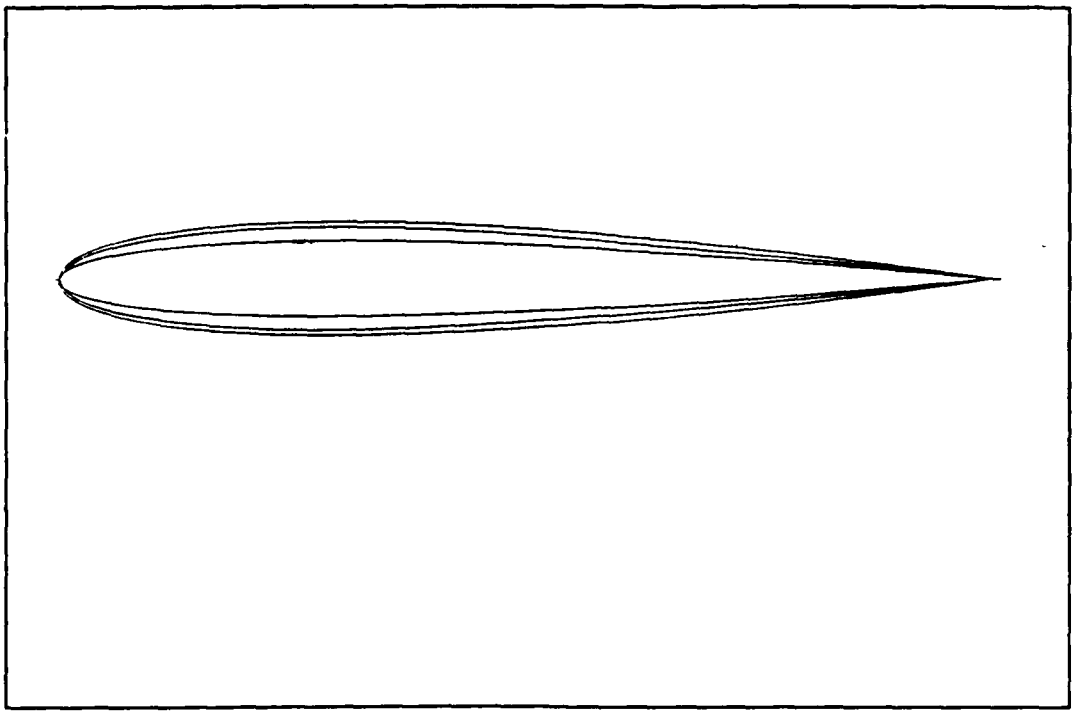

Figure 6.5 : Airfoil Shapes Using the Parallel Optimization Scheme for Test Case I

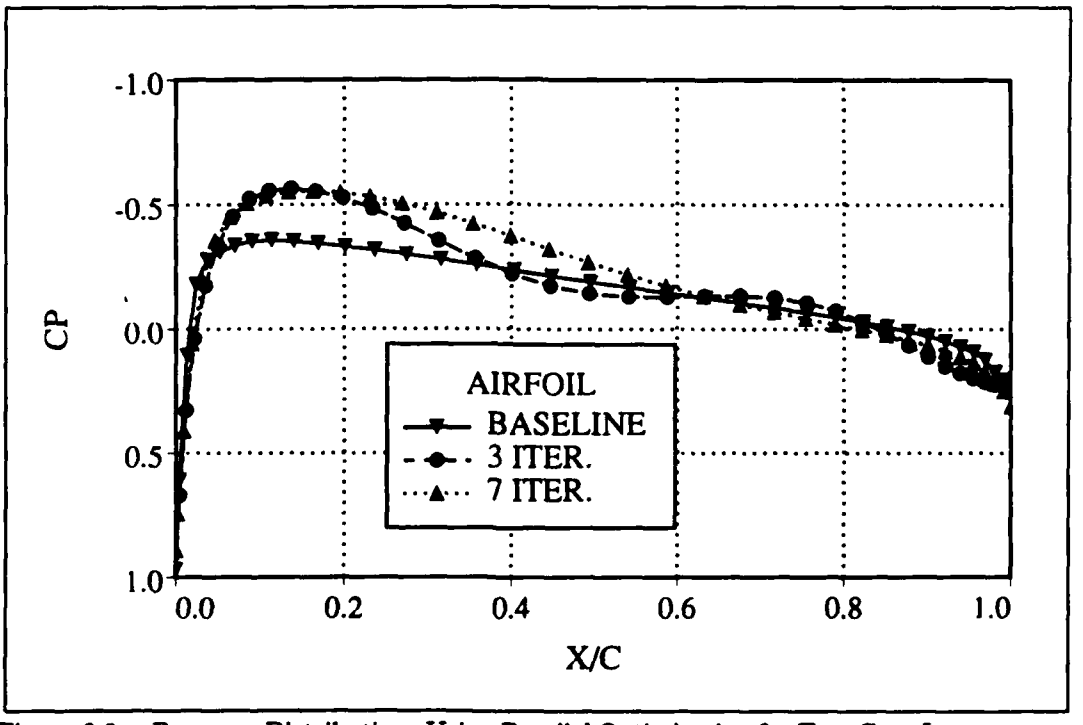

Figure 6.6 : Pressure Distributions Using Parallel Optimization for Test Case I

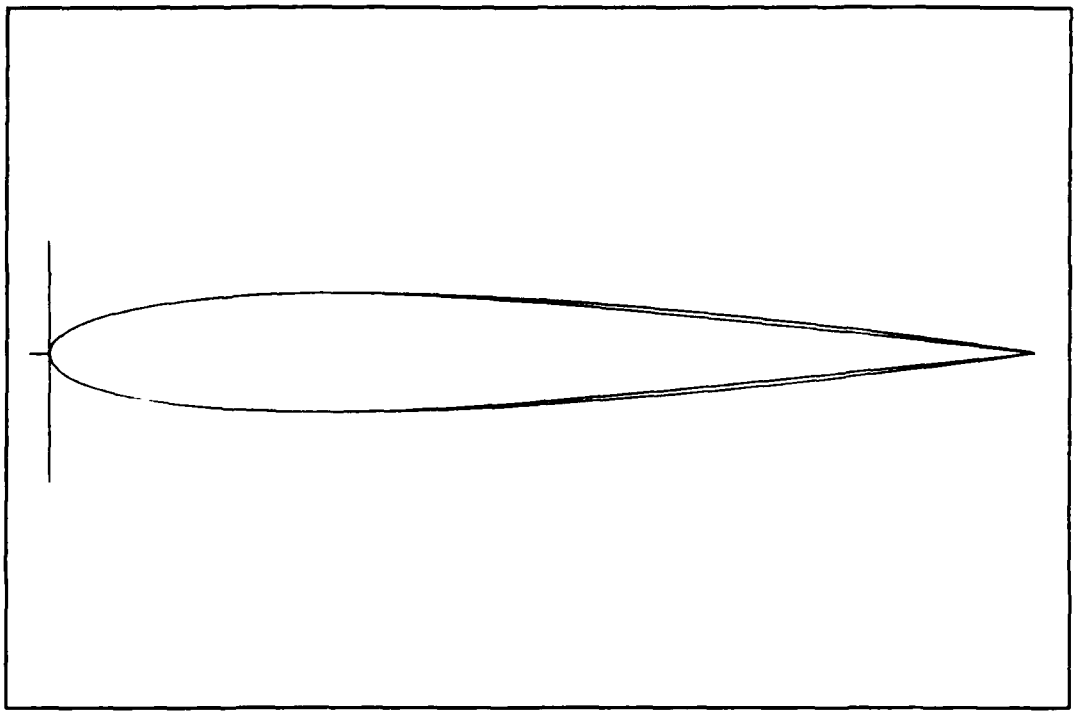

Figure 6.7 : Parallel Optimization Design and Target Airfoil Shapes for Test Case I

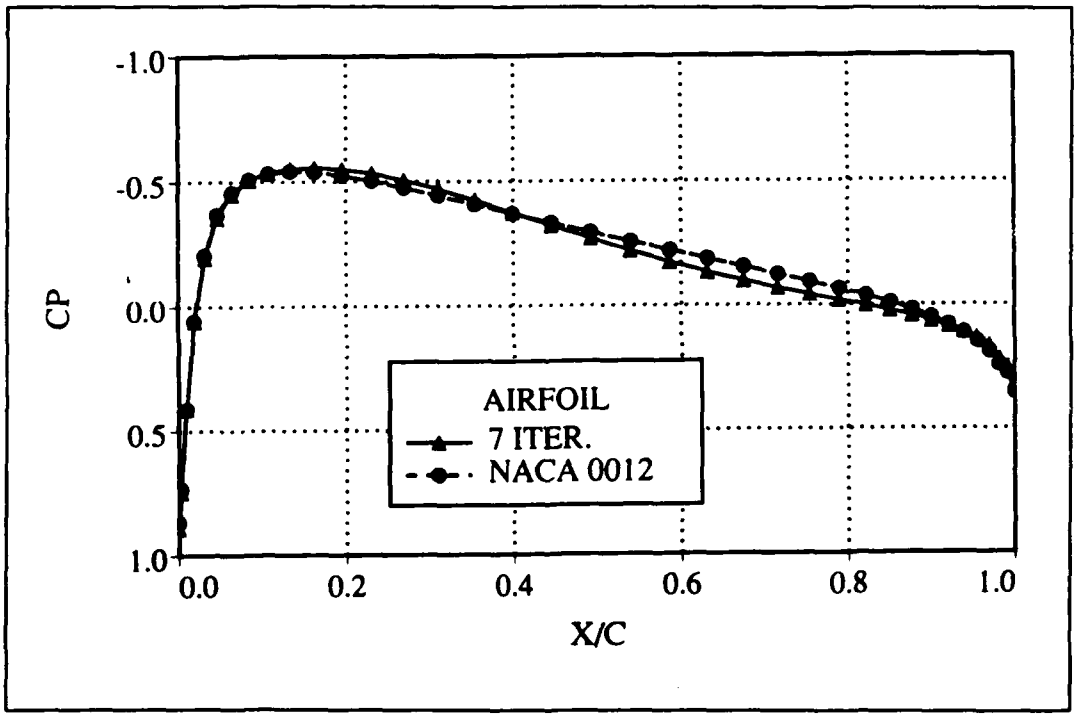

Figure 6.8 : Pressure Distributions for Parallel Design and Target Airfoil Shapes

8. Importance of Designer Intervention

Every airfoil design scheme requires a significant amount of intervention and supervision by the designer. The designer must select the design criteria, the mathematical formulation of the objective function, and the input variables for the flow solver and the optimization routine. The importance of user intervention in a design scheme is demonstrated by the designer's selection of the maximum variation of airfoil thickness for the directional searches in this test case. If a maximum variation was chosen to be a very small value, the convergence of a design solution is delayed because the optimum set of independent variables were not found in the directional search.

An example of this is shown in Table 6.2 where the maximum variation is shown using 0.5% chord and 1.0% chord. Selection of the smaller maximum variation of airfoil thickness increased the number of optimization cycles required. However, if too large of a maximum thickness variation was selected, impractical airfoil geometries would be calculated and result in unstable flow-field solutions. For this test case, 1% chord maximum variations proved the best whole percentage selection, and flow-field instabilities resulted with selections greater than 4%.

Table 6.2 : Selections of Maximum Thickness Variations
Baseline Airfoil : NACA 0008
Target Airfoil : NACA 0012
M = 0.6, AOA = 0 degrees

THICKNESS VARIATION	OPTIMIZATION CYCLES	FINAL OBJECTIVE	CPU TIME (HR:MIN:SEC)
0.5% CHORD	10	0.091	5:46:03
1.0% CHORD	7	0.064	4:01:20

9. Results from the Fully Newton Parallel Optimization Scheme

The fully Newton method optimization scheme was also applied to this test case. Because 8 independent variables were used to describe the shape of the airfoil, all 128 processors of the hypercube were used for the parallel calculations of estimations of the gradient vector and the Hessian matrix. Also, 128 evaluations of objective functions were calculated in parallel for each directional search.

The convergence history for the first seven cycles of the airfoil design using the parallel Newton and the parallel quasi-Newton optimization schemes are shown in Figure 6.9. The convergence rate of the parallel Newton optimization routine is much slower than for the parallel quasi-Newton optimization routine. Twice, the line searches of the parallel Newton optimization scheme failed to reduce the objective function, and local searches were required. The parallel quasi-Newton estimation of the Hessian matrix resulted in more effective line searches than the second-order accurate finite-difference computation of the full Hessian matrix.

10. Summary of Test Case 1

The parallel quasi-Newton optimization scheme proved superior to the sequential quasi-Newton and the parallel fully Newton schemes. The sequential version contained numerous inefficiencies due to its reliance upon forward-difference gradient estimations and parabolic interpolation used in directional searches. Also, the parallel scheme's utilization of multiple processors for parallel estimation of the gradient vector and for directional searches greatly decreases the time required to reach a solution. The parallel Newton method shows a much lower convergence rate than the parallel quasi-Newton scheme and is limited to only 8 independent variables for the 128 processor machine used. For the remaining test cases, only the parallel quasi-Newton optimization scheme is evaluated.

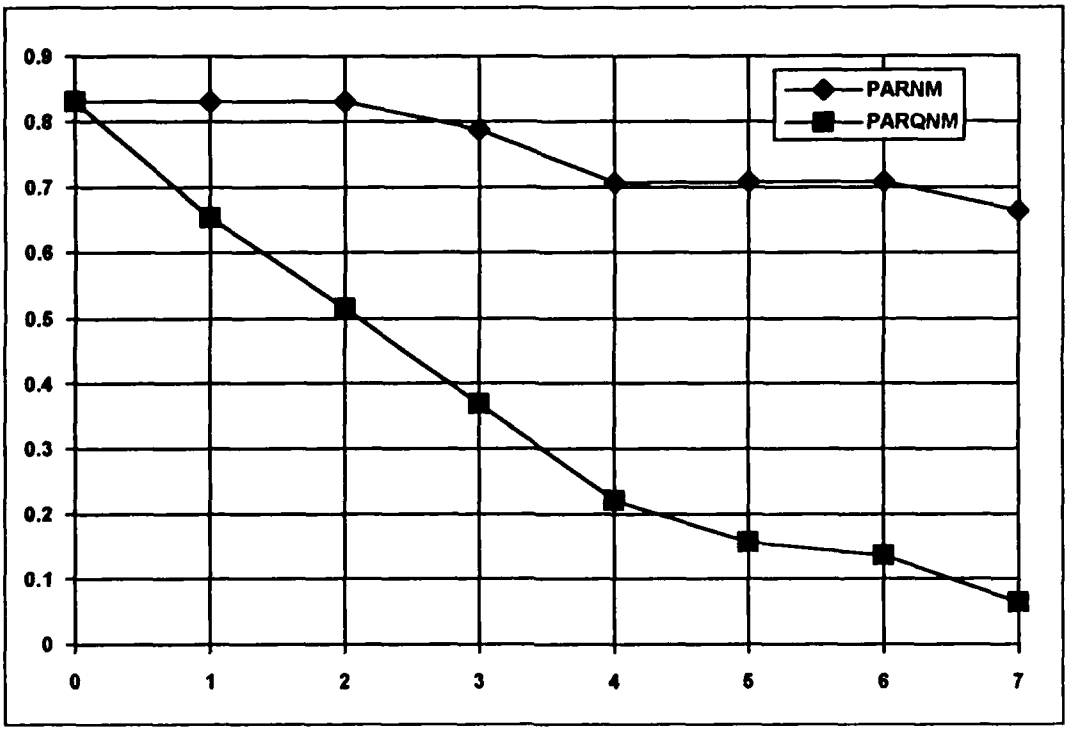

Figure 6.9 : Convergence Histories for Parallel Newton and Quasi-Newton Methods

C. TEST CASE 2 : DESIGN OF A LIFTING SUBSONIC AIRFOIL

A similar test case to the first was conducted for an airfoil generating lift in a subsonic flow-field. The second test case utilized only the parallel quasi-Newton optimization routine to design an airfoil to match the pressure distribution of a cambered airfoil at a small angle of attack. The efficiency of the parallel optimization design scheme is evaluated.

1. Baseline Airfoil

The baseline airfoil used in this test case is a NACA 1410. This airfoil has a maximum camber of 1% and a maximum thickness of 10%. The airfoil geometries involved in the second test case were more complex than those in the first because the camber and thickness of the airfoils were varied each optimization cycle.

2. Independent Variables

The independent variables used for the second test case were selected to be eight collocation points describing the surface of the airfoil. Four independent variables were collocation points on the lower surface of the airfoil, and four independent variables were collocation points on the upper surface of the airfoil. The McDonnell Douglas geometry package was used to compute two fifth-order Chebychev polynomials describing the surfaces. The collocation points were varied slightly each optimization cycle to recompute airfoil shapes needed for the gradient calculation and the directional search.

3. Grid Generation

The GRAPE subroutine was used to compute a 133 x 34 grid around each airfoil shape described by the independent variables.

4. Flow Solver

The inviscid pressure distributions around the airfoil shapes evaluated in the second test case were calculated using the explicit Euler solver RK2EULER. RK2EULER updated the flow-field properties around various airfoil shapes at an angle of attack of two degrees and a freestream Mach number of 0.6 to solve for their steady-state pressure distributions. The pressure at the surface points of the airfoil shapes were again used for the calculation of the objective function in the optimization process.

Since the flows around the airfoil shapes were not symmetric like those in the first test case, more iterations were assigned for the flow-field evaluations. Based upon the results of Chapter III, 1600 flow-field iterations were performed for each objective function evaluation. Also, the flow-field properties were initialized to

freestream conditions prior to every evaluation due to the more complicated flows than the first test case.

5. Performance Criterion

The performance criterion for the second test case was similar to that for the first. The goal of this test case was to design an airfoil to match or optimize the inviscid pressure distribution around a NACA 2412 airfoil at 2 degrees angle of attack and a Mach number of 0.6. The coefficients of pressure for 73 points around the NACA 2412 at the design flight conditions were read from an input file.

The Mach contours around the baseline airfoil are shown in Figure 6.10, and the Mach contours around a NACA 2412 in identical flight conditions are shown in Figure 6.11. Also, the corresponding pressure distributions around both airfoils are shown in Figure 6.12.

6. Stopping Criterion

The stopping criterion was simply set as six optimization cycles based upon estimated processing time.

7. Results

The results of the second test case are similar to those of the first. The test case was completed using 16 processors and 8 hours of processing time on the hypercube. More processing time was required than with the first test case due to the more flow-field iterations required for each flow-field evaluation.

The convergence history of this case is shown in Figure 6.13. The airfoil design application reduced the objective function to less than 10% of its original value in only five iterations.

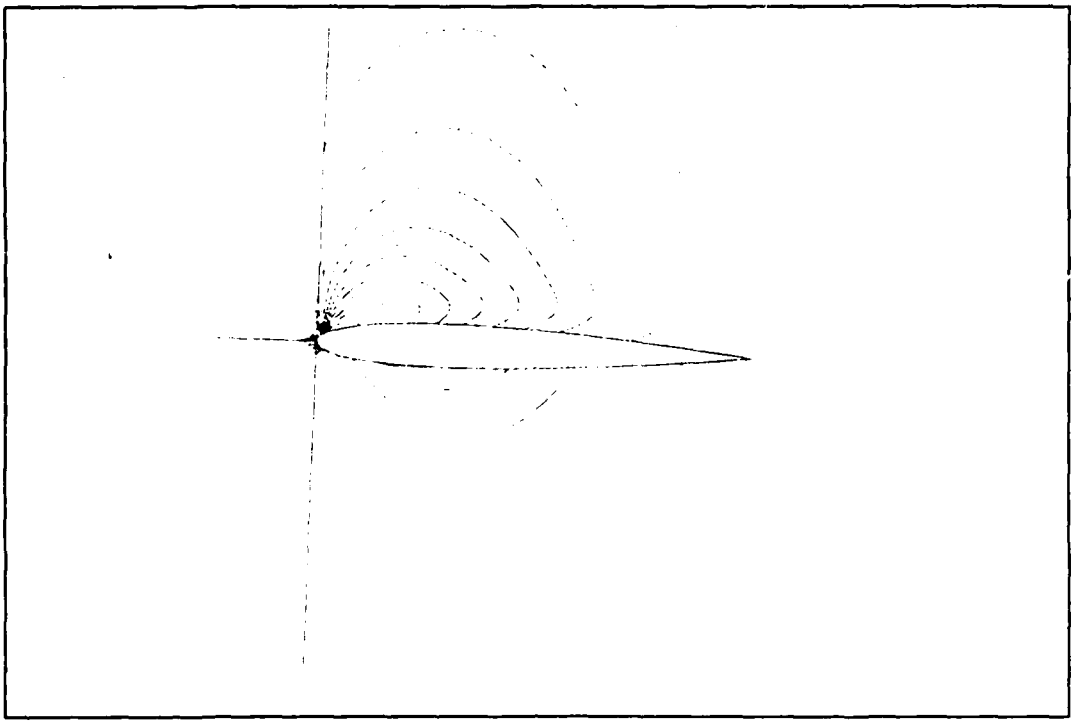

Figure 6.10 : Mach Contours Around Baseline Airfoil for Test Case 2

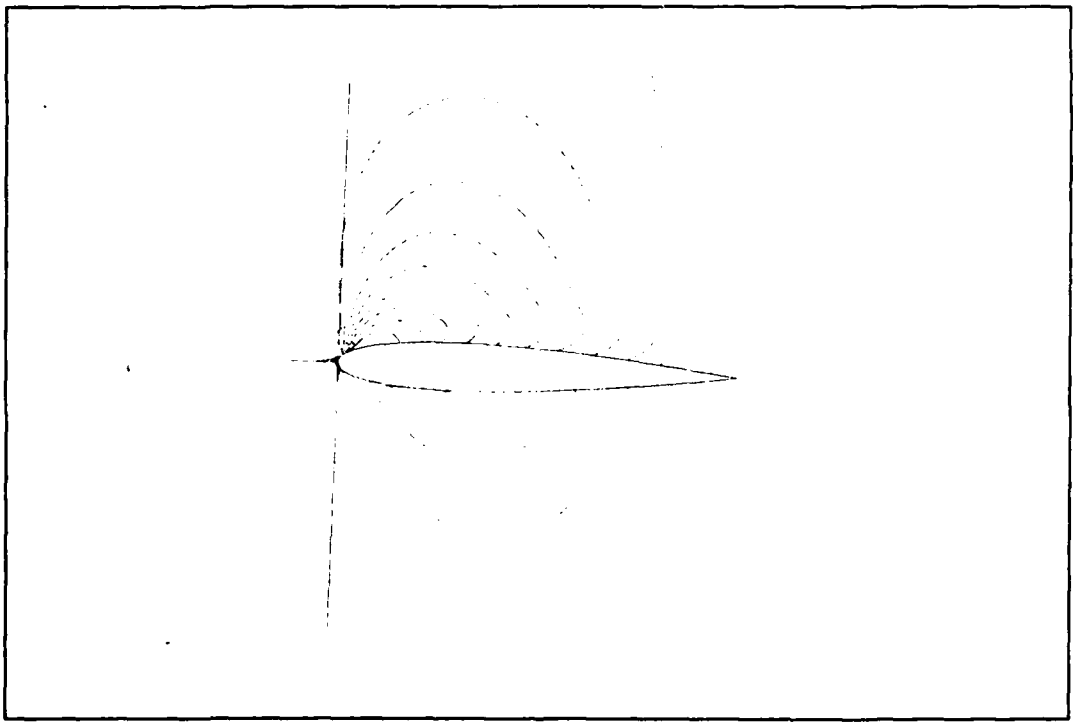

Figure 6.11 : Mach Contours Around Target Airfoil for Test Case 2

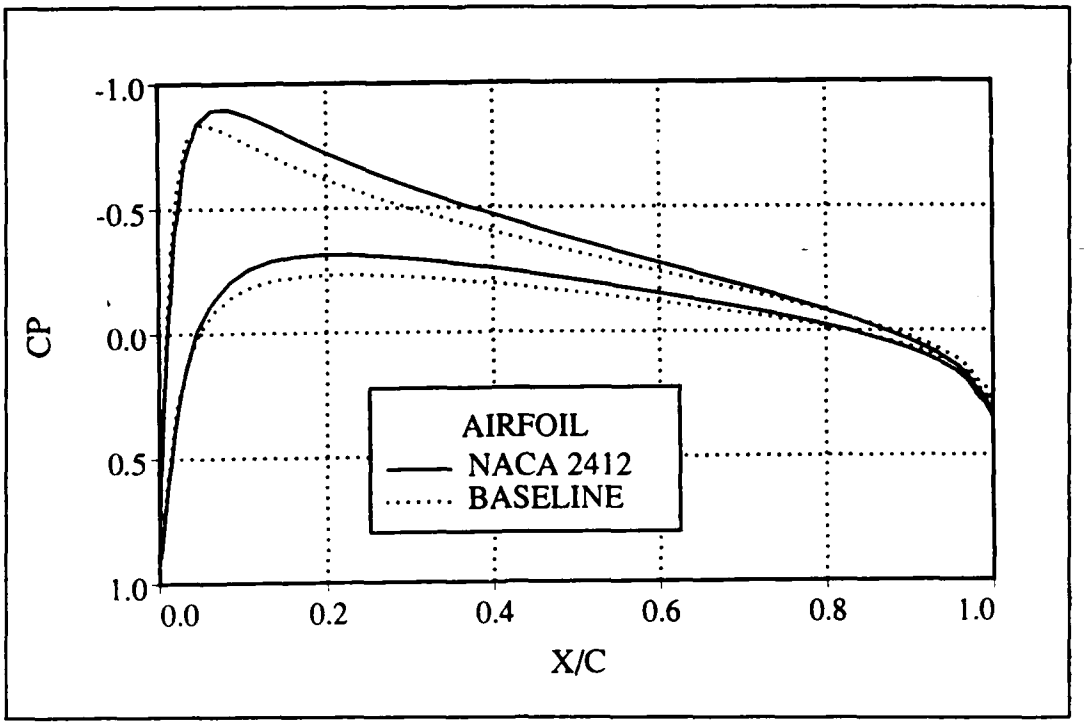

Figure 6.12: Pressure Distribution Around Baseline and Target Airfoils

Figure 6.13 : Convergence History for Test Case 2

Mach contours around the optimization design airfoil solution are shown in Figure 6.14. The design and target airfoil shapes and their pressure distributions are compared in Figure 6.15 and Figure 6.16 respectively. The airfoil shapes and their resulting pressure distributions are similar. This test case demonstrates the parallel quasi-Newton optimization scheme's capability of designing more complicated airfoil geometries.

8. User Intervention

In order to obtain a solution for this test case, the flow-field must be initialized based upon freestream conditions and a density of one prior to each flow-field evaluation. When the flow-field was not re-initialized to freestream conditions such as in the first test case, the solution would not converge to match the target pressure distribution and airfoil shape.

D. TRANSONIC AIRFOIL DESIGN

A parallel design application was performed to maximize the lift-to-drag ratio of an airfoil in transonic flight conditions. In transonic flow, a small change in the shape of an airfoil results in a large change in its pressure distribution due to changes in shock locations. Unlike the previous test cases, a specific airfoil shape was not used as a target for the solution.

1. Baseline Airfoil and Independent Variables

The baseline airfoil used in this test case was a NACA 0012. Eight independent variables were chosen as the thicknesses at eight points along the chord of the airfoil. The McDonnell Douglas geometry package was used to compute a ninth-order Chebychev polynomial describing the thickness distribution of the airfoil. The camber along the airfoil was set to zero and not varied.

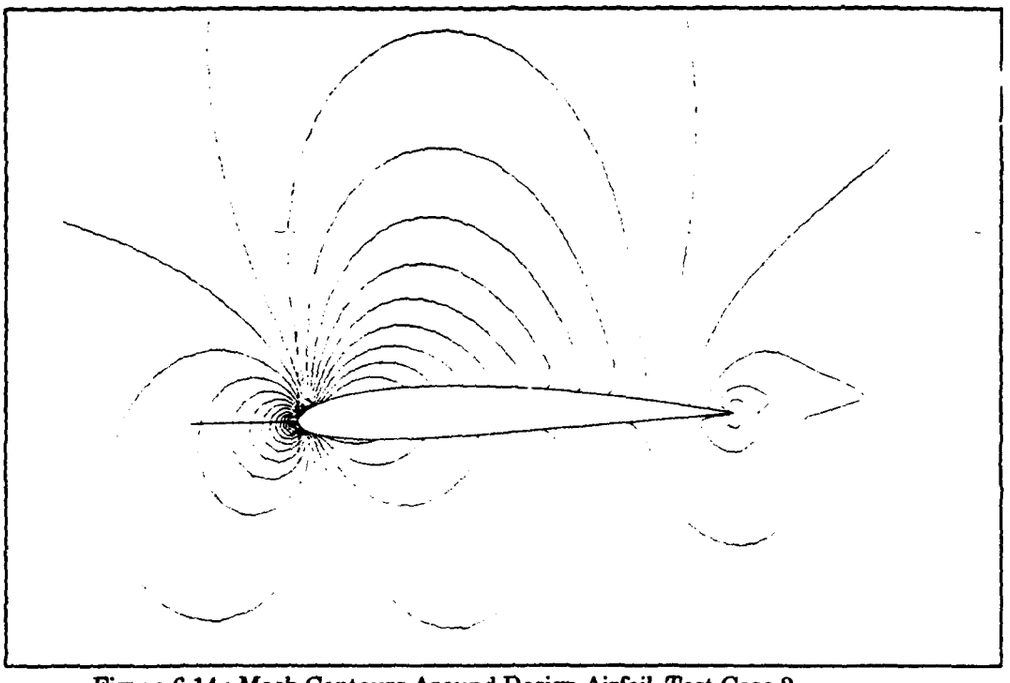

Figure 6.14 : Mach Contours Around Design Airfoil, Test Case 2

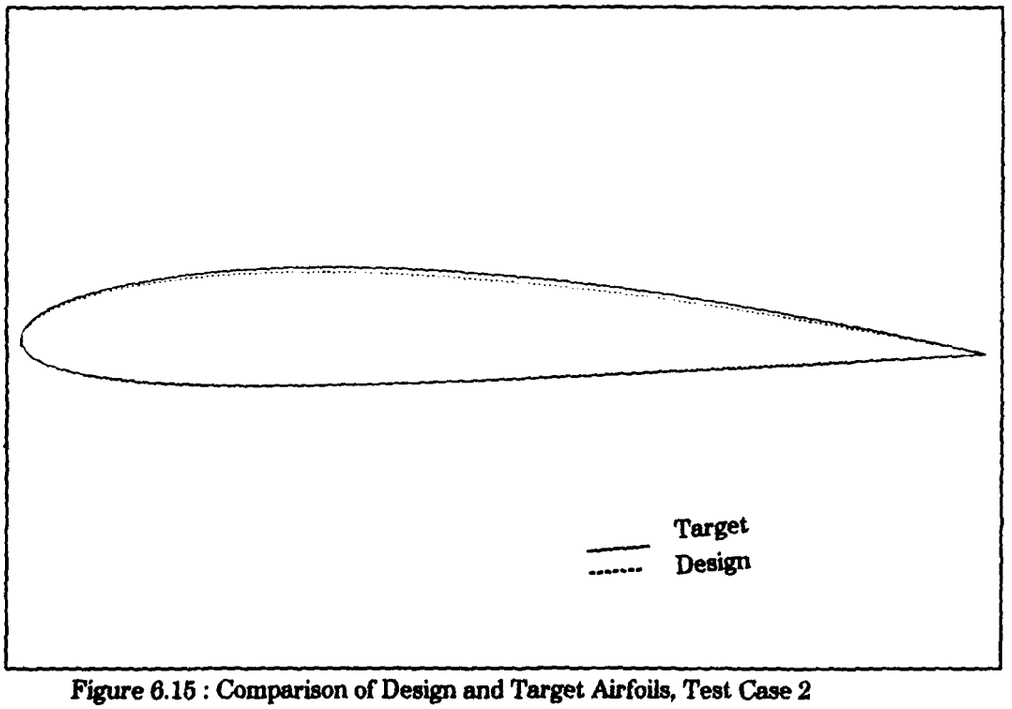

Figure 6.15 : Comparison of Design and Target Airfoils, Test Case 2

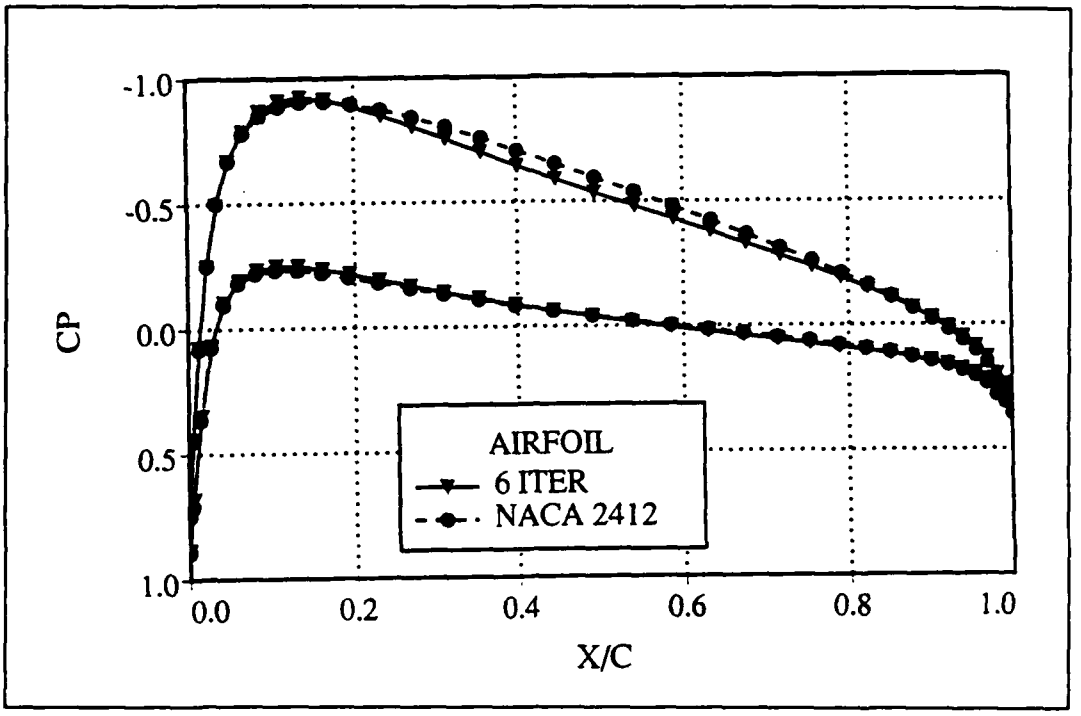

Figure 6.16 : Pressure Distributions for Design and Target Airfoils, Test Case 2

3. Grid Generation

The GRAPE subroutine was again used to compute a 133 x 34 grid around each airfoil shape described by the independent variables.

4. Flow Solver

The flow-field around each evaluated airfoil shape was calculated using the explicit Euler solver RK2EULER. RK2EULER updated the flow-field properties around various airfoil shapes at an angle of attack of one-half degree and a freestream Mach number of 0.8 to solve for their steady-state pressure distributions. The evaluated lift and drag coefficients for each geometry were used in the calculation of the objective function.

Similar to the second test case, flow-field properties were initialized to freestream conditions prior to each evaluation, and 1800 flow-field iterations were performed due to the presence of shocks in the flows.

5. Performance Criteria

The goal of this application was to design a symmetric airfoil to maximize its inviscid lift-to-drag ratio. The objective function to be minimized was selected as the square of the inviscid drag-to-lift ratio:

$$f = \left(\frac{C_{dw}}{C_l} \right)^2 .$$

The Mach contours around the baseline airfoil are shown in Figure 6.17.

6. Stopping Criterion

Three optimization cycles were set for this design application.

7. Results

Mach contours around the design airfoil are shown in Figures 6.18. The geometries and the pressure distributions of the baseline and design airfoils are shown in Figure 6.19 and Figure 6.20. The thickness of the design airfoil is more evenly distributed than with the NACA 0012. Subsequently the shock on the design airfoil is weaker and farther aft.

The convergence history of this design application is shown in Figure 6.21. Unlike the first two test cases where the objective function would equal zero if the geometry of the design and target airfoils matched exactly, the minimum theoretical objective function was not known. The parallel optimization routine reduced the objective function to 30% of its original value after three iterations. Also, the

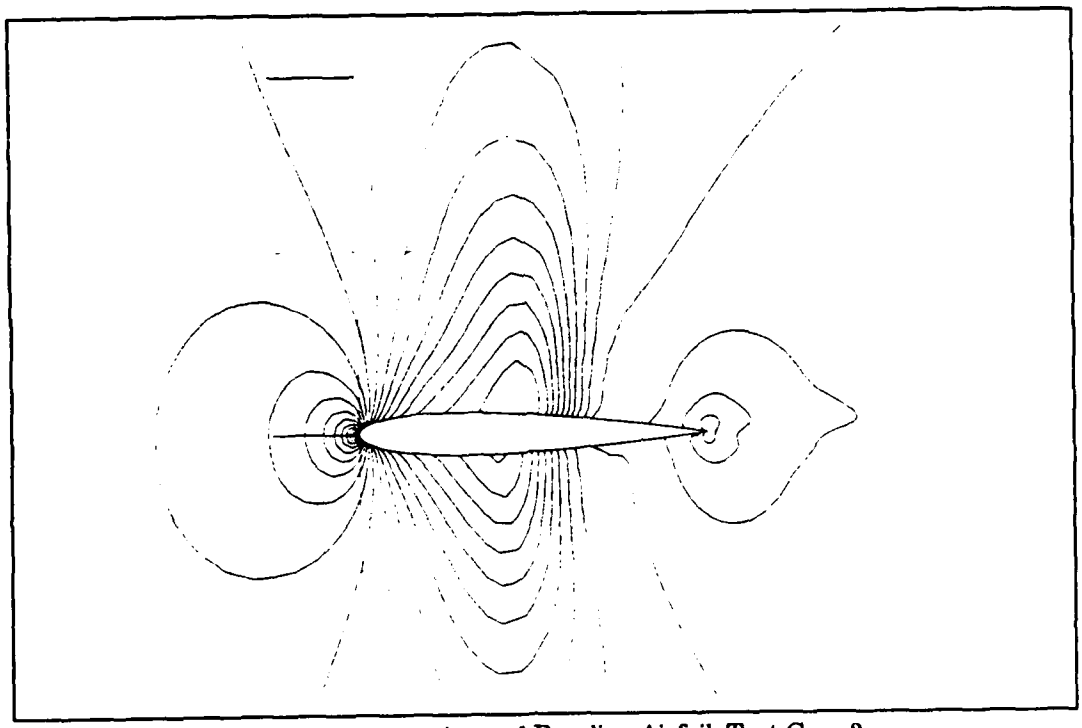

Figure 6.17 : Mach Contours Around Baseline Airfoil, Test Case 3

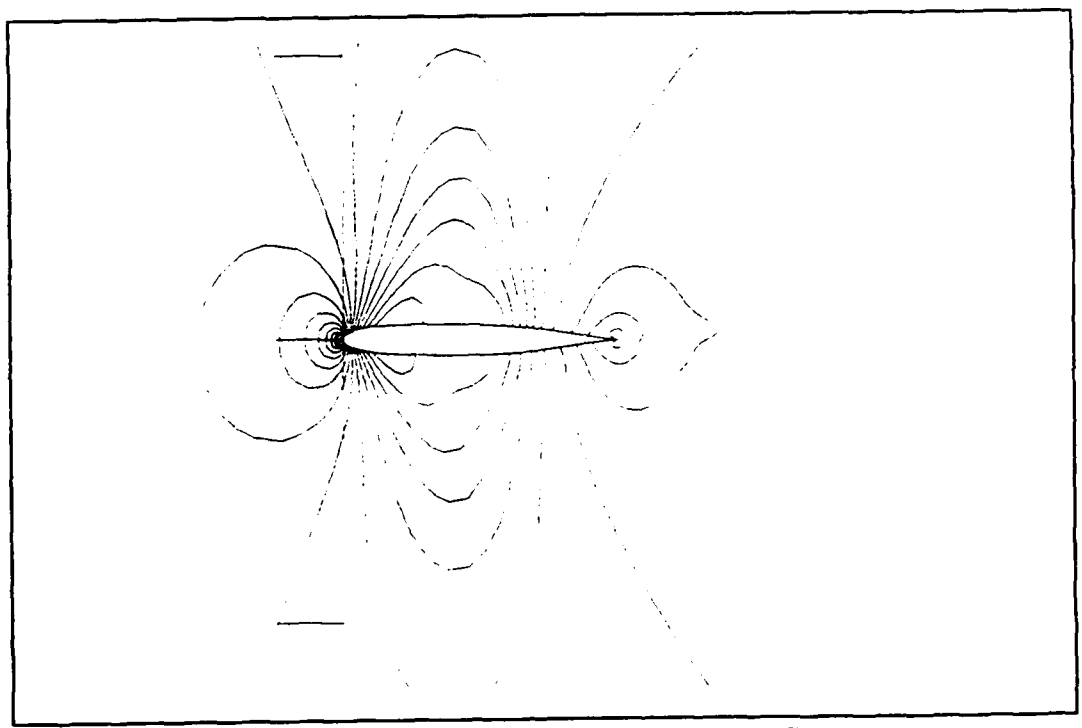

Figure 6.18 : Mach Contours Around Design Airfoil, Test Case 4

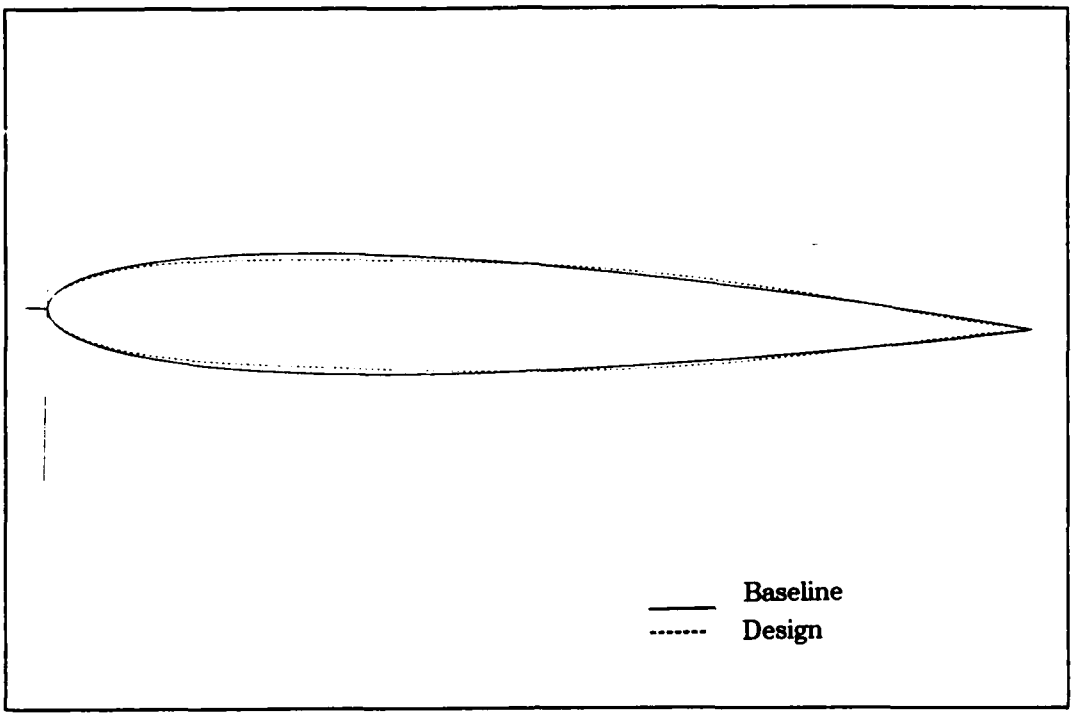

Figure 6.19 : Comparison of Baseline and Design Airfoils, Test Case 3

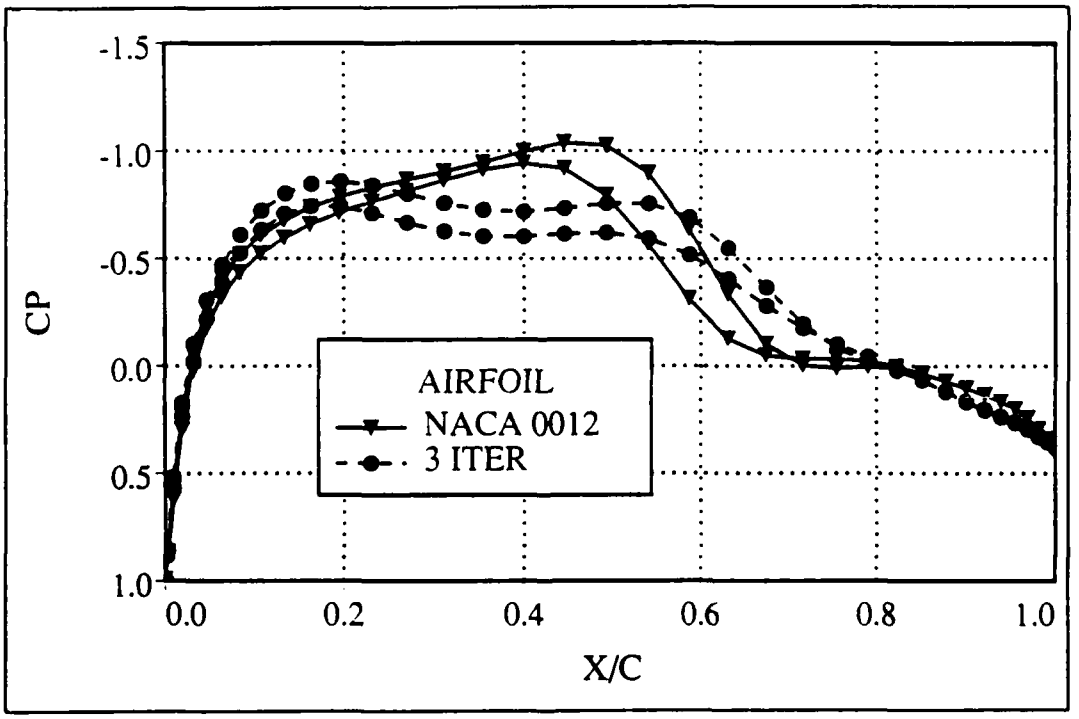

Figure 6.20 : Comparison of Baseline and Design Pressure Distributions

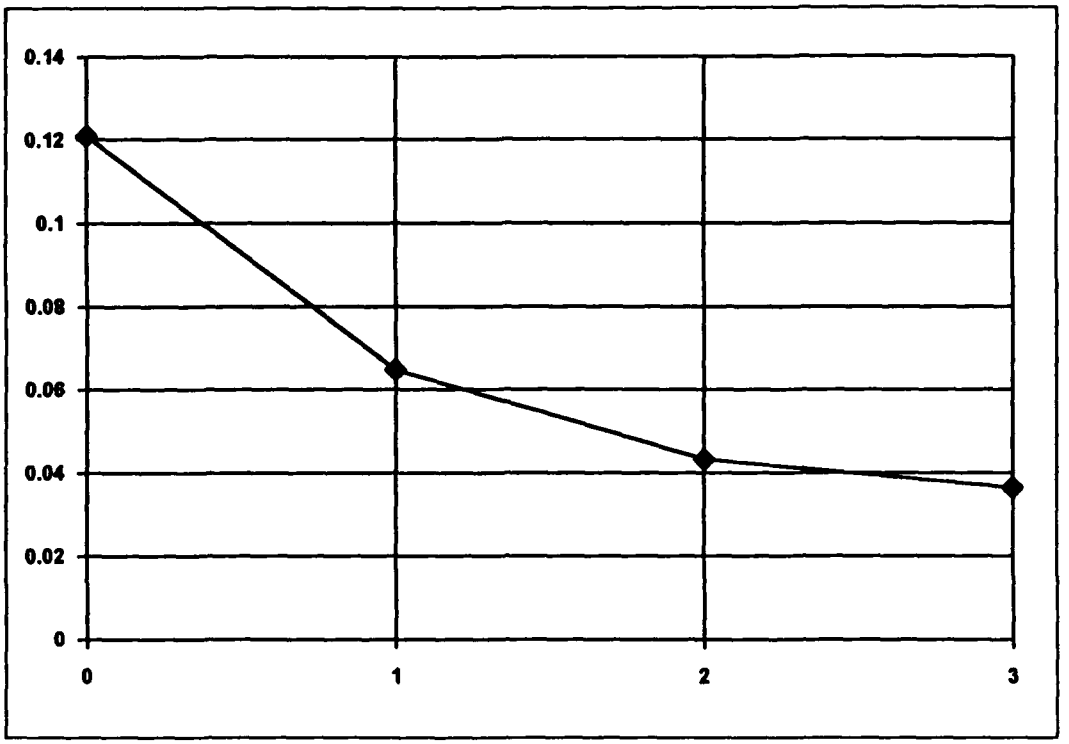

Figure 6.21 : Convergence History for Test Case 3

convergence rate of the routine decreased significantly with the final two iterations as the objective function approached a minimum. The test case was completed using 16 processors and 7 hours of processing time on the hypercube.

E. CASCADE BLADE DESIGN

The objective of the final test case was to design a two-dimensional symmetric cascade blade shape to minimize viscous losses while maintaining adequate volume to allow for such concepts as cooling of the blade. Most turbomachinery blade design relies upon subsonic analyses or transonic potential analysis which may not provide the best analyses of the internal flow-field. This test demonstrates the practicality of utilizing an efficient Navier-Stokes flow solver with the parallel quasi-Newton optimization scheme for aerodynamic design.

This test case has several variations from previous ones. Unlike the previous external aerodynamic design test cases, flows in turbomachinery are highly rotational and can be dominated by shock waves and viscous effects. Since the design criteria is based upon viscous losses, a Navier-Stokes flow solver is used. Also, this application involved internal rather than external flow, and different boundary conditions must be applied. These variations demonstrate the versatility of using an optimization routine for the design of airfoils or cascade blades for numerous performance criteria.

1. Baseline Airfoil and Independent Variables

The baseline cascade blade used for this application was a symmetric NACA 0012 airfoil. Eight independent variables were chosen to represent the thickness at eight points along the chord of the cascade blade. The McDonnell Douglas geometry package was used to calculate the coefficients of a ninth-order Chebychev polynomial which described the thickness distribution of each cascade blade shape evaluated in the design process.

2. Grid Generation

A modification to the GRAPE grid generation program was used to generate a 250 x 60 C type grid around the airfoils evaluated in the design process. This version of GRAPE was modified to allow a more general clustering of points around the leading and trailing edges of turbomachinery blades and to improve the generation of periodic boundaries in blade rows. Grids used for Navier-Stokes flow solvers require more grid points than those used with inviscid flow solvers, especially near the surface of the airfoil for calculation of flow-field properties within the boundary layer where viscous effects are significant. The number of grid points was

chosen based upon previous experience with the Navier-Stokes flow solver. Figure 6.22 shows the grid around the baseline compressor blade row.

3. Flow Solver

The selected performance criteria required a Navier-Stokes flow solver to evaluate the viscous losses of the internal cascade flow. A multi-stage Runge-Kutta scheme flow solver with a Baldwin-Lomax turbulence model was chosen.

Chima [12] developed an explicit algorithm for quasi three-dimensional flows in turbomachinery. This efficient Navier-Stokes code was developed for turbomachinery design and analysis. The flow solver is easily vectorizable because of its explicit scheme and uses both variable time steps and implicit residual smoothing. The flow solver allows for many user inputs for the application including the number of Runge-Kutta stages to be used and whether the flow is inviscid or viscous.

a. Governing Equations

The two-dimensional thin-layer unsteady Navier-Stokes equation in conservative form for an arbitrary coordinate system can be written as follows :

$$\partial_t \hat{q} + \partial_\xi \hat{E} + \partial_\eta \left(\hat{F} - \mathrm{Re}^{-1} \hat{S} \right) = \mathbf{0} \qquad (6.1),$$

where

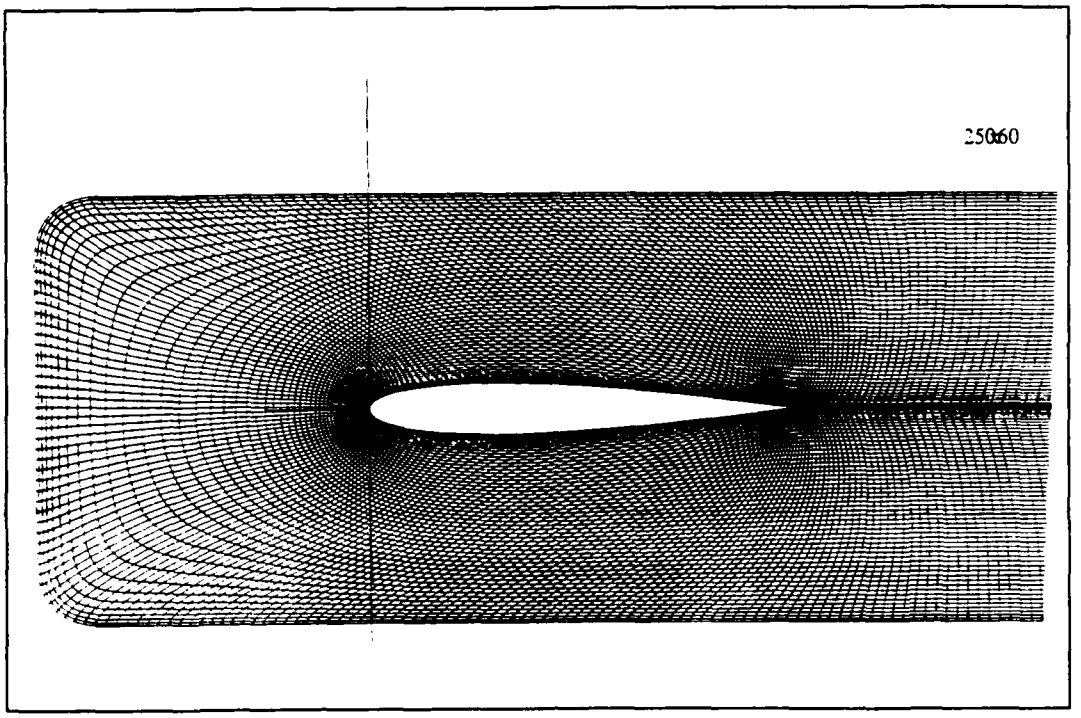

Figure 6.22 : 250 x 60 Grid Around Baseline Airfoil, Test Case 4

$$\hat{q} = J^{-1}\begin{bmatrix}\rho\\ \rho u\\ \rho v\\ e\end{bmatrix}, \quad \hat{E} = J^{-1}\begin{bmatrix}\rho U\\ \rho uU + \xi_x p\\ \rho vU + \xi_y p\\ (e+p)U\end{bmatrix}, \quad \text{and} \quad \hat{F} = J^{-1}\begin{bmatrix}\rho V\\ \rho uV + \eta_x p\\ \rho vV + \eta_y p\\ (e+p)V\end{bmatrix}.$$

The viscous flux vector is written as

$$\hat{S} = J^{-1}\mu \begin{bmatrix} 0 \\ C_1\partial_\eta u + C_2\partial_\eta v \\ C_2\partial_\eta u + C_3\partial_\eta v \\ C_4\partial_\eta\left[\frac{e}{p} - \frac{1}{2}\left(u^2+v^2\right)\right] \\ \quad +\left(C_1 u + C_2 v\right)\partial_\eta u + \left(C_2 u + C_3 v\right)\partial_\eta v \end{bmatrix},$$

where

$$C_1 = \frac{4}{3}\eta_x{}^2 + \eta_y{}^2, \quad C_2 = \frac{1}{3}\eta_x\eta_y, \quad C_3 = \eta_x{}^2 + \frac{4}{3}\eta_y{}^2, \text{ and } C_4 = \frac{\gamma}{\mathrm{Pr}}\left(\eta_x{}^2 + \eta_y{}^2\right).$$

The contravariant velocities are defined using the metrics to be

$$U = \xi_x u + \xi_y v$$
$$V = \eta_x u + \eta_y v.$$

These equations are nondimensionalized using reference quantities and assume the specific heat and Prandtl number, Pr, are constant. The effective viscosity may be written as

$$\mu = \mu_{\text{laminar}} + \mu_{\text{turbulent}} \qquad (6.2).$$

The thin viscous layer assumption has been invoked to eliminate the streamwise viscous derivatives, which reduces the processing time and allows for the

computation of separated flows. The algebraic two layer eddy-viscosity model developed by Baldwin and Lomax [17] is used for the evaluation of turbulent flows. The turbulence model is applied to an expanding C shaped region that expands downstream from the leading edge and covers the entire wake region.

b. Boundary Conditions

For boundary conditions, the total pressure, total temperature and flow angle are specified. The velocity at the inlet is extrapolated from the interior. Also, the inlet density and pressure are calculated from isentropic relationships. For viscous flows, the velocity components are set equal to zero on the surface. Surface pressure and a specified wall temperature are used to calculate surface densities. The overall total to static pressure ratio is also specified which fixes the back pressure.

c. Multistage Algorithms

A five-stage Runge-Kutta scheme is applied to this problem with the level of time integration being user selectable. Residuals and dissipative terms are calculated at each point in delta form and added to the previous values of the flow-field properties every iteration.

Artificial dissipation is added to prevent odd-even point decoupling and to allow shock capturing. Second-order dissipation is added to prevent pre-shock oscillations and is based upon density rather than pressure for computational efficiency.

d. Variable Time Step

A spatially variable time step is used to accelerate the convergence of the fine grid algorithm to steady-state. The time step at each point in the grid is calculated using a constant Courant number.

e. Flow-field Evaluations

Because of the complexity of the flow, the flow-field around each cascade blade geometry was initialized based upon freestream conditions prior to each performance evaluation. The larger grid size, more Runge-Kutta steps and viscous calculations require significantly more processing time per flow-field iteration than was needed using RK2EULER in the previous test cases. Therefore, 400 flow-field iterations were performed for each geometry evaluated. Mach contours around the baseline airfoil are shown in Figure 6.23.

4. Performance Criteria

The purpose of this design application was to design a cascade blade which minimizes viscous losses while maintaining adequate volume for the blade. An objective function was formulated which accounted for a trade-off of these two factors.

The explicit Navier-Stokes flow solver was used to evaluate a loss coefficient for the viscous losses through the cascade. The loss coefficient, C_{loss}, was based upon the loss of total pressure from the cascade inlet to the cascade exit due to viscous effects.

For a decrease in viscous losses, the symmetric cascade blade would decrease in thickness and volume. The change in airfoil volume from its original volume was incorporated in the calculation of the objective function, f, with

$$f = C_{loss} \frac{Vol_{NACA\ 0012}}{Vol_{DESIGN\ BLADE}},$$

where Vol is the volume of the cascade blade.

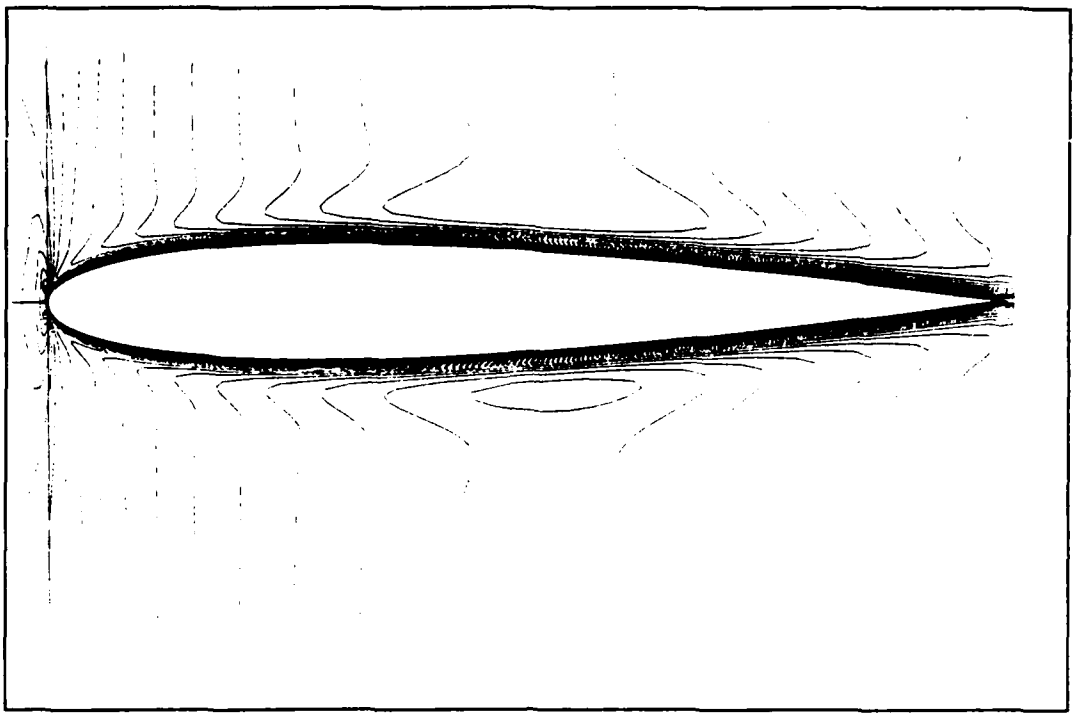

Figure 6.23 : Mach Contours Around Baseline Cascade Blade

5. Stopping Criteria

The stopping criteria was set for 2 optimization cycles because of the increased processing time required for the viscous flow evaluations. Each optimization cycle requires approximately 4 hours processing time on 16 processors.

6. Results

The parallel optimization routine successfully reduced the objective function. Two optimization cycles were completed using 16 processors of the iPSC/i860 parallel computer. Mach contours around the design airfoil are shown in Figure 6.24. The baseline and design geometries are compared in Figure 6.25. Also, the values of airfoil volume, coefficient of viscous losses and objective function each optimization cycle are shown in Figure 6.26.

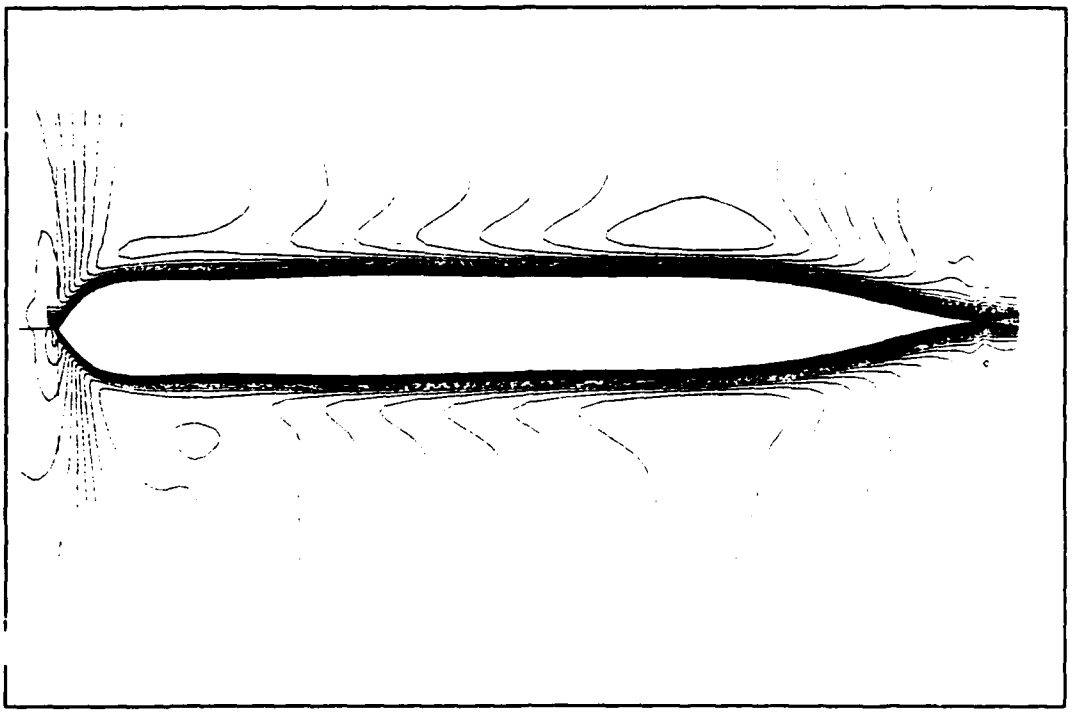

Figure 6.24 : Mach Contours Around Design Cascade Blade

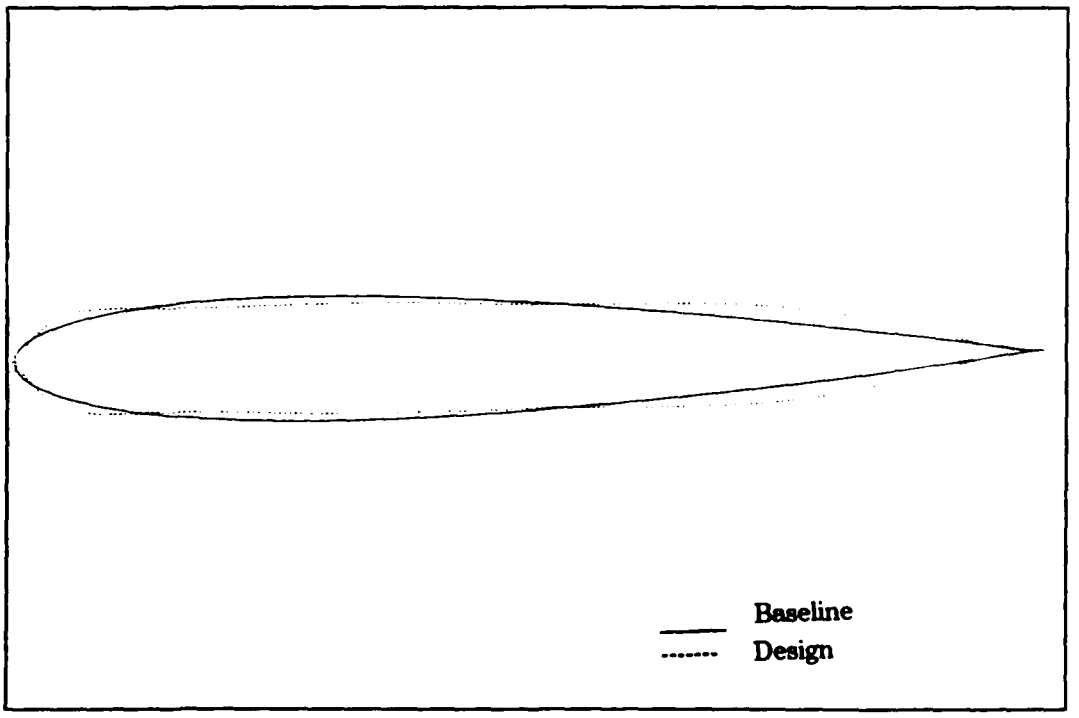

Figure 6.25 : Comparison of Baseline and Design Cascade Blades

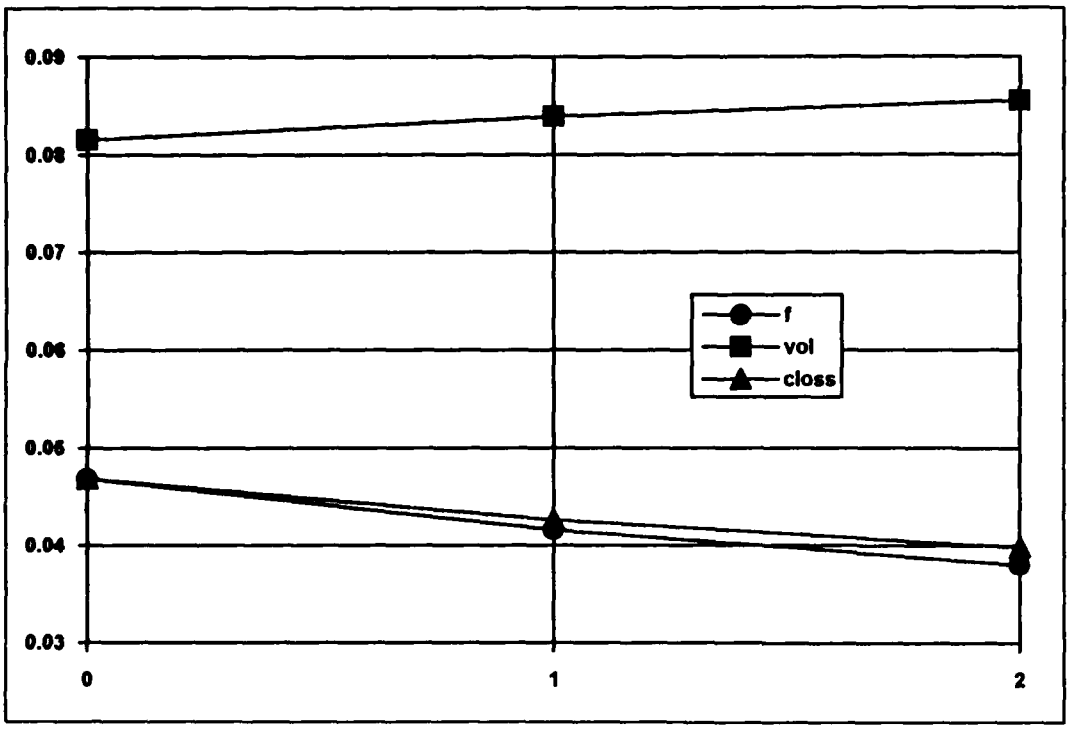

Figure 6.26 : Convergence History for Cascade Blade Design

The optimization scheme decreased the thickness of the airfoil near its nose and increased its thickness aft of 60% chord. The flow over the design airfoil accelerated more gradually which resulted in a smaller wake. The overall volume of the airfoil increased 5%, and the viscous loss coefficient decreased 19% in the design process.

7. Observations

The viscous design test case was simplified to demonstrate the capability for aerodynamic optimization involving both internal and external viscous flows. Numerous variations to this design problem are possible including changes to the type and number of independent variables, geometric constraints and nature of the objective function. The utilization of explicit schemes and parallel processors allows

solutions to optimization problems involving viscous flows to be reached in reasonable time periods.

VII. SUMMARY AND CONCLUSIONS

This research is the first to demonstrate the successful utilization of a parallel supercomputer and a Navier-Stokes flow solver in aerodynamic design. Many topics were investigated including the advantages of explicit flow solvers and of a fully Newton optimization scheme utilizing parallel processors for airfoil design.

Airfoil design via optimization techniques requires intensive CFD solutions over different airfoil geometries. The aerodynamic performance of these various shapes are evaluated to search for a shape which optimizes the desired performance criteria. The utilization of explicit flow solvers and multiple processors for the evaluation of the aerodynamic performance of various shapes can greatly decrease the processing time required for an airfoil design application.

A. SUMMARY

A two-step Runge-Kutta scheme inviscid flow solver was developed and compared to a similar Crank-Nicholson scheme flow solver. For subsonic and transonic test cases, the two-step Runge-Kutta scheme flow solver solved for the steady-state pressure distribution around an airfoil roughly five times faster than the Crank-Nicholson scheme.

A superior quasi-Newton optimization scheme utilizing parallel processors and a fully Newton optimization scheme were developed and utilized for airfoil design. The parallel quasi-Newton optimization scheme was used to design an airfoil to match the inviscid pressure distribution of a symmetric airfoil shape over an order of magnitude faster and more efficiently than a quasi-Newton optimization scheme

using an identical processor. The fully Newton parallel optimization scheme was not as successful as the parallel quasi-Newton scheme.

The parallel quasi-Newton optimization routine was further utilized in another test case and two design applications. The camber and thickness of an airfoil were varied to match the inviscid pressure distribution of a known lifting airfoil. Also, a symmetric airfoil in inviscid transonic flow was designed to maximize its lift-to-drag ratio. Furthermore, a Navier-Stokes flow solver was coupled with the parallel quasi-Newton optimization scheme to design a cascade blade to minimize viscous losses and retain sufficient volume for cooling purposes.

B. CONCLUSIONS

1. Advantages of Explicit Flow Solvers

Explicit scheme flow solvers are capable of solving for the steady-state flow-field around an airfoil many times faster than implicit or Crank-Nicholson scheme flow solvers. Implicit and Crank-Nicholson scheme flow solvers require inversions of large matrices to update the properties at each point in the flow-field simultaneously. Explicit scheme flow solvers such as the two-step Runge-Kutta scheme flow solver developed for this research are more easily vectorizable than the implicit or Crank-Nicholson schemes and require no matrix inversions.

The vast majority of processing time required for airfoil design via optimization is used calculating the aerodynamic performance of different airfoil shapes. Therefore, utilization of an explicit flow solver with an optimization scheme can greatly decrease the required processing time for an airfoil design application.

2. Parallel Optimization Schemes

Parallel processing machines such as the Intel hypercube parallel supercomputer can be used to greatly increase the speed and the efficiency of airfoil design via optimization. For airfoil design via optimization, expensive objective functions based upon CFD solutions are calculated to describe the performance of the airfoil. Parallel processors are used during different stages of the optimization cycle to evaluate the performance of multiple airfoil shapes simultaneously. These parallel performance calculations greatly decrease the time required for airfoil design.

a. Parallel Quasi-Newton Method

The parallel quasi-Newton optimization routine developed in this research is superior to a similar sequential version. The utilization of parallel processors for second-order accurate gradient vector estimations and in directional searches to minimize the objective function increases the efficiency of the quasi-Newton routine. For the particular case of airfoil design via optimization which requires multiple calculations of expensive objective functions, the utilization of the parallel quasi-Newton routine can result in design solutions in hours instead of days using the sequential version.

b. Parallel Fully Newton Method

The fully Newton method optimization scheme developed in this research is not as effective and requires numerous more processors than the parallel quasi-Newton scheme. Variations to the scheme may prove its worthiness in future optimization applications.

3. Optimization Utilizing a Navier-Stokes Flow Solver

A major advantage of airfoil design via optimization over inverse airfoil design techniques is the independence of the optimization routine from the flow solver. This allows various flow solvers to be used with the same optimization routine, and the flow solver can be selected based upon the desired performance criteria which may vary each problem.

In the past, Navier-Stokes flow solvers have not been used in aerodynamic design due to the large amount of required processing time. This restriction in the choice of flow solvers has also restricted the designer's selection of desired performance criteria. This research has proven that successful design applications involving viscous phenomena can be accomplished in a reasonable amount of processing time utilizing an efficient explicit Navier-Stokes flow solver and the parallel quasi-Newton optimization scheme.

4. Importance of Designer Intervention

The single most important factor in any aerodynamic design application is the supervision and intervention of the designer in a design process. The designer must decide the desired performance criteria to optimize and mathematically formulate this criteria into an objective function. The designer must also select the appropriate flow solver and independent variables for the optimization routine based upon the performance criteria. Furthermore, the designer must carefully examine the results and make necessary changes to the problem in order to ensure a meaningful solution.

C. FUTURE WORK

The subject of aerodynamic design involves multiple disciplines including aerodynamics, CFD, computer science, and optimization techniques. A single person could spend his entire life conducting research any one of these fields. Some suggestions for future work in these areas as they relate to aerodynamic design are presented.

1. Flow Solvers

The two-step Runge-Kutta Euler flow solver developed for airfoil design solved the steady-state pressure distribution many times faster than a similar Crank-Nicholson scheme but also showed a decreased convergence rate after higher numbers of iterations were completed. This is primarily due to the low CFL stability limitations for the two-step Runge-Kutta scheme. The overall convergence history of the inviscid flow solver may be improved using a four-step Runge-Kutta scheme and residual smoothing to increase the allowable CFL. The addition of more stages to the Runge-Kutta scheme may also increase the processing time required to reach a solution depending upon the stopping criteria selected. The number of stages used for a Runge-Kutta flow solver must be considered for each application.

2. Parallel Programming

The utilization of parallel processors greatly decreases the required processing time in an airfoil design problem by solving multiple objective functions corresponding to different airfoil shapes in parallel. These calculated objective functions are passed between processors for the estimation of the gradient vector and for line searches to locate a minimum. Simultaneous internode message passing, global messages and global operations are used to minimize the

communication time, but new techniques further reducing the communication time could be employed.

3. Optimization Techniques

Optimization techniques are continually being updated and evaluated. Any optimization routine involving the evaluation of expensive multivariable objective functions can probably use parallel processors advantageously to reduce the required processing time.

The parallel quasi-Newton optimization routine successfully utilizes parallel processors for the gradient calculations and for directional searches. When the directional search fails to locate a minimum, a local search including two parallel line searches is performed to find a set of independent variables which corresponds to a lower objective function. Different methods of local searches should be investigated to determine one which best employs the parallel processors.

Variations of the fully Newton method parallel optimization program are worthy of future evaluations. One reason that the parallel Newton optimization routine did not perform as well as the quasi-Newton one may be the simplification used to ensure that the estimated Hessian was positive definite. A parallel Newton optimization routine may prove more efficient and faster than the parallel quasi-Newton routine. Also, a fully Newton optimization routine may not require a line search after the gradient vector and Hessian matrix estimations are performed.

4. Design Applications

Many areas of aerodynamics could benefit using parallel optimization design applications, especially areas with little empirical data. Hypersonic wing design and helicopter rotor design may be ideal applications if the desired performance is formulated correctly in the objective function. Also, design of compressor and

turbine blades utilizing Navier-Stokes flow solvers may prove to be successful endeavors.

The primary purpose of this research was to demonstrate the advantages of using a parallel computer with multiple vector processors for airfoil optimization. It seemed logical to apply the relatively new technology of parallel supercomputing to an intuitively parallel problem. Future advances in computer technology should be evaluated and applied to airfoil optimization and other areas of science.

REFERENCES

1. Lighthill, M. J., *A New Method of Two Dimensional Aerodynamic Design*, ARC, Rand M 2112, 1945.

2. Jameson, A., *Aerodynamic Design Via Control Theory*, Journal of Scientific Computing, Vol. 3, 1988, pp233-260.

3. Giles, M. B. and Drela, M., *Two-Dimensional Transonic Aerodynamic Design Method*, AIAA Journal Vol. 25, No. 9, 1987.

4. Campbell, R. L. and Smith, L. A., *A Hybrid Algorithm for Transonic Airfoil and Wing Design*, AIAA-87-2552-CP, 1987.

5. Bock, K. W., *Aerodynamic Design By Optimization*, AGARD Conference Proceedings No. 463, 1989.

6. Vanderplaats, G.N., *CONMIN - FORTRAN Program for Constrained Function Minimization*, NASA TMX-62.282, 1973.

7. Sanger, N. L., *The Use of Optimization Techniques to Design Controlled Diffusion Compressor Blading*, NASA Technical Memorandum 82763, 1982.

8. Kennelly, R. A., *Improved Method for Transonic Airfoil Design-by-Optimization*, AIAA-83-1864, 1983.

9. Miel, G., *Supercomputers and CFD*, Aerospace America, January 1992.

10. Merkle, C. L., *Computational Fluid Dynamics of Inviscid and High Reynolds Number Flows*, Department of Mechanical Enginerring, Pennsylvania State University at University Park.

11. Ekaterinaris, J. A., Clarkson, J. D., and Platzer, M. F., *Computational Investigation of Airfoil Stall Flutter*, 6th International Symposium on Unsteady Aerodynamics and Aeroelasticity of Turbomachines and Propellers, 1991.

12. Chima, R. V., *Inviscid and Viscous Flows in Cascades with an Explicit Multi-Grid Algorithm*, AIAA Journal, Vol. 23, No. 10, 1986.

13. Sorenson, R. L., *A Computer Program to Generate Two-Dimensional Grids About Airfoils and Other Shapes by the Use of Poission's Equation*, NASA TM-81198, 1980.

14. Gill, P. E., and Murray, W., *Quasi-Newton Methods for Unconstrained Optimization*, J. Inst. Maths. Applics. 9, 1972.

15. Verhoff, A., Stookesberry, D. and Cain, A., *An Efficient Approach to Optimal Aerodynamic Design, Part 1: Analytic Geometry and Aerodynamic Sensitivities,* AIAA 93-0099, January 1993.

16. Forsythe, G. E., and others, *Computer Methods for Mathematical Computations* , Englewood Cliffs : Prentice Hall, 1977.

17. Baldwin, B. S. and Lomax, H., *Thin -Layered Approximation and Algebraic Model for Separated Turbulent Flows*, AIAA paper 78-257, Jan. 1978.

INITIAL DISTRIBUTION LIST

1.	Library, Code 52 Naval Postgraduate School Monterey, CA 93943-5000	2
2.	Mr. Tom Lawrence Naval Air Systems Command Headquarters AIR-53011 Washington, DC 20381	1
3.	Associate Professor Garth V. Hobson Department of Aeronautics and Astronautics Code AA/Hg Naval Postgraduate School Monterey, CA 93943-5000	2
4.	Professor Daniel J. Collins Department of Aeronautics and Astronautics Code AA/Co Naval Postgraduate School Monterey, CA 93943-5000	1
5.	Professor Oscar Biblarz Department of Aeronautics and Astronautics Code AA/Bi Naval Postgraduate School Monterey, CA 93943-5000	1
6.	Professor Harold A. Titus Department of Electrical Engineering Naval Postgraduate School Monterey, CA 93943-5000	1

7. Professor Beny Neta — 1
   Department of Mathematics
   Code MA/Nd
   Naval Postgraduate School
   Monterey, CA 93943-5000

8. Dr. Augustus Verhoff — 1
   McDonnell Aircraft Company
   Mail Code 034-1260
   St Louis, MO 63166

9. LT Stephen C. Brawley — 3
   1655 Harbor Drive
   Vista, CA 92083

10. Defense Technical Information Center — 2
    Cameron Station
    Alexandria, VA 22304-6145